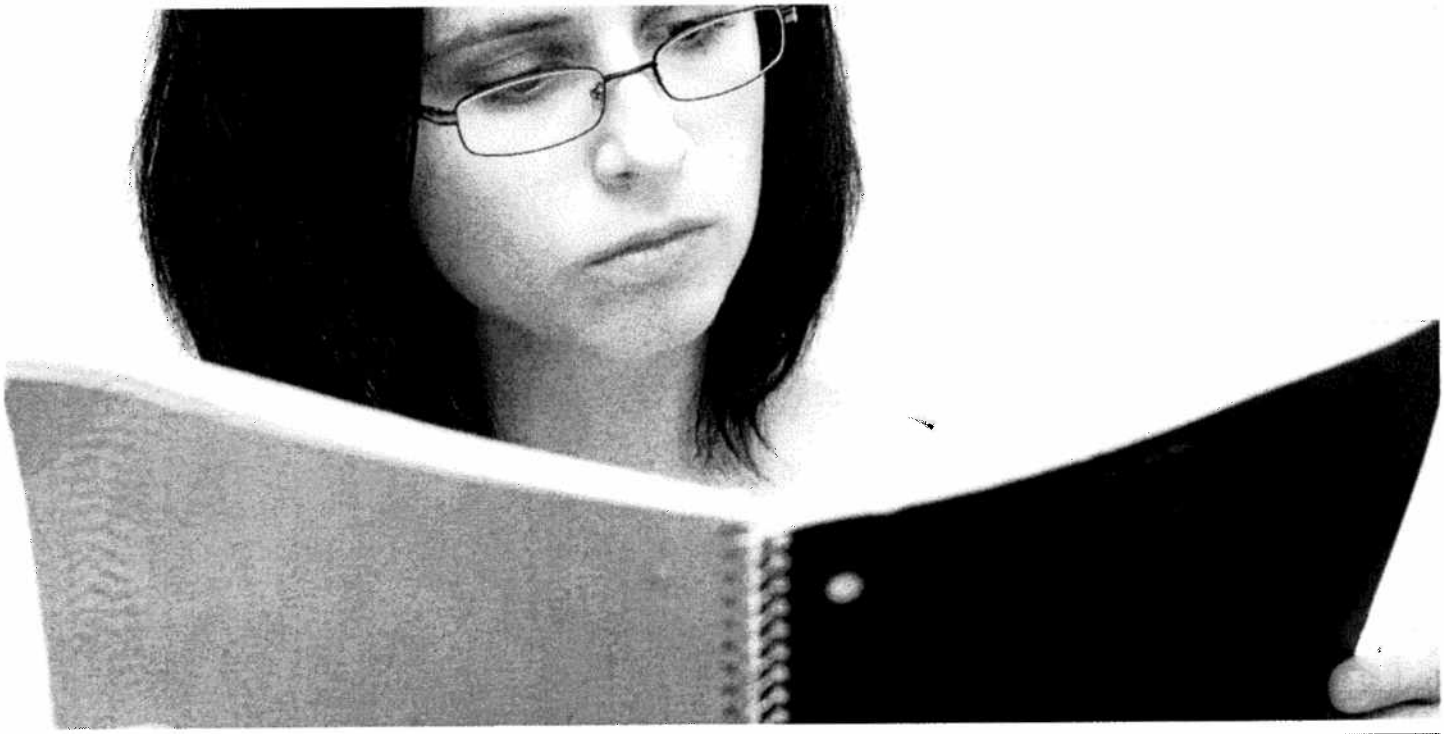
Exemplar work

GCE Computing COMP4



Candidate 7

Please note: every effort has been made to respect the right of anonymity of the source of this project and of those who have contributed to it, without compromising the work or invalidating it for the purpose to which it is to be put. Some personal details of contributors may therefore remain. We ask the reader to respect their right to privacy.

Note:

The page numbers in the separate document, *Commentary on Project 7*, refer to the page numbers given in the *bottom centre* of this Exemplar Project 7, *not* those to the bottom right hand side.

Contents

Section 1

Analysis

Background to and identification of the problem

[redacted] School is a school based in Southampton that teaches pupils from the ages 11-18 years old.

The following procedures were under taken in order to gain knowledge of the current solution and also to investigate what the problem is and what the end user would like to have on the new system:

- Interview
- Observation of the current system and any limitations that might need to be taken into consideration when developing the new system.
- Examination of current paper work undertaken such as the print job form used to see how the accounts are presented.

Below shows a copy of the interview questions and answers that were undertaken when learning more about he current system and the problem to be solved.

Interview Questions

1. How do you complete the task at the moment?
   - Manually by hand
   - Taking data from handwritten form
   - Each user has a user number
   - Printer holds all the information if it is a simple job (no finishing)
   - Anything complex is done

2. What data do you have to put in to the current system?
   - All print job requests
   - Spreadsheet

3. What information do you get out of the current system?
   - Spreadsheet of costs
   - No. of copies

4. What format is the output from the system?
   - Excel sheet

5. What are the good points of the current system?
   - Simple to use

6. What are the problems with the current system?
   - Takes a lot of time
   - Jobs can sometimes get lost
   - Has everything been answered correctly
   - Charging to the correct department someone might put the wrong one

7. What features would like the new system to have?
   - An online print job form that can be filled by the user
   - Internal email system

8. What has prompted you to consider having a new system?
   - Inefficiencies
   - Time taken to do things

9. Who will be using the system and what is their level of expertise?
   - Teachers
   - Pupils
   - Management access

10. What hardware do you have at present?
    - Printers
    - Computers
    - Specification of computer printers

As gathered from the observation undertaken at the moment in order for a department to request something to be photocopied or printed, they have to fill out a 'Print job form' by hand and then this would be handed over to the reprographics which will process the job. The 'Print job form' contains details such name, number of copies wanted, finishing details and colour of paper wanted.

This means that the entire print job requests are manually processed by hand. Every print job request is handwritten. This can take a lot of time to do and sometimes things can get out of hand when you have lots of print job requests given to you which increases the risk of the reprographics losing it through the piles of paper.

Also each time someone fills out a print job form, the reprographics first of all checks to see if it is a valid job as some jobs for example if they want a quantity of above 700 pages then it should be processed externally. However if it is accepted then the reprographics submits the information manually on a spreadsheet showing the total cost of print, what has been print and how many copies want to be done.

The problem with the current system is that it can take a lot of time to process each print job request, and an even more delay to when print job request come all at the same time. This is because each print job request has to be processed manually one at a time for things like to see if everything has been entered correctly. There are some limit to the number of copies that can be done at one time, for example if a department has requested for over 800 pages, then this job has to be done externally as it is more cheap and efficient. This again adds to the time taken to process each print job form and with lots of requests to deal with this adds to the problem. Also the chance for print job forms increases as there are lots of orders to deal with and this would make the system less efficient as important jobs that have strict deadlines might not even be processed because it has got lost somehow.

Each job has to then be submitted onto a spreadsheet, details such as the cost of printing, the costs for using different types of papers (i.e. card) and special jobs (i.e. laminate, binded). With the build up of lots of print job requests sometimes there can be a risk that different jobs can be charged to different departments. This reduces the reliability of the system.

Proposed solution (summarised)

The reprographics teacher has asked if this problem can be solved by having all the tasks done on a computer system, a programme where people can fill out the print job request online and then this is sent onto the reprographics computer system. This way everything is more organised and less time consuming as all requests are done systematically. Also the computer validates the print job form before it is sent to the reprographics and at the same time updating the account file automatically.

At the same time when the online form (print job form) has been submitted, the cost of the printing will be entered into a separate file named 'accounts' and this is where the reprographics teacher will be able to print a copy of it to give to the schools account. The accounts file will include details such as the user ID, date, what they have printed, and the number of copies and the total cost of the printing. This will be an improvement to the current system as in the current system this is currently done manually where the reprographics would look at the print job forms and submit the required information onto a spreadsheet. This was very time consuming, especially when there were a lot of print job forms to handle which made it a very inefficient way to handle things.

In addition, once the user submits the online form (print job form) the credit for that user is updated automatically, this will be based on the number of copies the user has requested to print.

Input forms, output forms, report formats from existing system

Input

- Member of department hands in a print job form

Processes

- Reprographics does the job
- Enter details of cost onto spreadsheet
- Saves a copy of the print job form for future reference

Output

- Print job request is completed and given to member of staff
- Spreadsheet printed at the end of each month and given to the school's accounts office

Algorithms used by current system that must be applied to the data in the new system

The current system is done manually by hand and no programming or algorithm has been used for the current system

Proposed solution (Detailed)

Each task that the programme does should be done on separate classes, so that later if there is a problem. It can easily be understood how the coding is done and what each class does.

All data that needs to be saved will be stored as an XML file. There should be a file for each function of the programme, for example, for the accounts file there should be one file and there should also be one file for the print job forms.

Every screen should be validated so that all the required details are under taken. Also all screens should have the option to either go back to the previous screen or go back to the main menu.

Login Screen

At the moment no-one has login details as the system is done manually where the user fills out a print job form on a piece of paper.

The new system should have a login screen where users can login using details that are provided by the reprographics. The new system should also allow the reprographic to add new users, remove existing users and also change user's passwords. The new system should allow everyone who uses the system to have the option of revealing their passwords by going through some security procedures like answering the answer to their security question that they used when signing up with the reprographics. 'Forgotten password' function should be included on the login screen.Further details on these functions are discussed below. This

Every user including the reprographics will have to sign into the system using a unique username which in this case will be the user ID and a password.

The following list was received from the reprographics which contains a list of the departments (users) and their User ID:

Department / User ID

D.OF E-2202

Drama/TS 2203

Art-2204

Biology-2205

Careers 2206

Chemistry-2207

Classics-2208

Ex. Studies-2209

Economics-2210

English 2211

Exams 2212

General Studies2213

French-2214

Geography-2216

German-2217

Junior Science-2218

DT-2219

History-2220

Mathematics-2223

Library 2222

Music-2226

Physics-2228

P.E (Games)-2229

PE Academic 2230

RE/RS-2231

ICT-2233

PHSE.-2235

Spanish-2236

Philosophy-2255

Oes 6610

Romania/Charities 6630

Sodexo 7710

Change password

Every user should be able to change their password. The reprographics will be able to change all user departments and also to view the passwords if the user has forgotten their password.

For the user, there should be two sections that need to be verified before the password is to be revealed to them.

The first section should be a page where the user has to enter the userID that they want the password for and then the user has to press an 'OK' button to confirm, once the 'OK' button has been pressed the system should check to see if the userID entered is a valid UserID and if it is a message box should be outputted saying 'UserID verified, please enter your security answer', the user should then be directed to the second section. If the UserID entered is not available then a message should be outputted saying 'Invalid UserID, please either try again or see the reprographics for further assistance', the user should then be redirected to the logging screen.

The second section should be designed so that the user's security question is outputted and then below it should be an area where the user can input their security answer. They then should have a button saying 'Reveal password', once the 'Reveal password' button has been pressed the system should check to see if the security question and security answer match if it does a message box should be outputted saying 'Your password is: (password of user should be shown here)', the user should then be directed to the login screen. If the security question and answer don't match then a message should be outputted saying 'Invalid security answer entered, please either try again or see the reprographics for further assistance', the user should then be redirected to the second section again.

Add a Department User ID (Only available to the reprographics)

This page should allow the reprographics to add a new User ID. Details that need to be collected on this page in order to create a user profile are:

1. User ID
2. Department
3. Password
4. Confirm Password
5. Security question
6. Security answer

Print job form

The central piece of the current system the print job form. Currently if the user would fill out thei job details on a piece of paper, and the reprographics does all the processing, like updating their print credits, updating the accounts and processing the job itself.

The reprographics wants the new system wants this to be all done on the computer including filling out the form, the following details need to be collected on the print job form in order to allow the reprographics to process the job, the following have to be filled in by the user in order to process the job and should be validated so that if they have not filled in the required details the job should not be allowed to be processed:

- ✓ UserID
- ✓ Staff initials
- ✓ Date the want the job completed (the date needs to be validatd so they enter in dd/mm/yyyy format)
- ✓ Copy supplied as (use of radio buttons is recommended for this)
- ✓ Paper type (drop down list recommended)

- ✓ Paper size (drop down list recommended)
- ✓ Paper colour (drop down list recommended)
- ✓ Number of copies (numeric counter feature recommended)

The following need to be including on the print job form as well however they need to be a check box feature as they are optional requirements that the user might want:

- ✓ Collated
- ✓ Back to back
- ✓ Stapled
- ✓ Wiro binding
- ✓ Laminate
- ✓ Glue bound
- ✓ Transparencies
- ✓ Two hole punched
- ✓ Folded
- ✓ Proof required

As soon as the submit key is pressed on the print job form, the following events must happen:

- Validation is run to check to see if it checks the entire requirement such as the number of prints is below 700 copies, the name of department entered is valid, all the spaces that are required are filled. Examples are the type of colour paper they want is stated, any extra features is select and if they don't want any extra feature they have selected the box to say that they don't want any extra features
- After the validation is done and the form is ok, the total cost of the print job is calculated, this will be done using opening a file that I will created that will contain the cost of the things, this file will be used to calculated the total cost of the print job and will automatically be added to the accounts page. This will also update the accounts page, stating the user id, number of copies printed, details of any extra features used, total cost of job, and also at the bottom of the sheet will have the total overall cost of the accounts.
- In addition after the valuation is done, the form will be saved as an xml file so that there is permanent storage of the data. The system should be designed so that once a job is submitted it is automatically saved onto the xml file.

The print job form should be added to a prioritised queue once submitted which can then be viewed by the reprographics, it will be prioritised by the date that they want the job completed by. Once the job is complete the reprographics has the option to remove the job from the queue, bringing the job behind it to the front of the queue.

Send Message

Will consist of a form that allows the user department to send a message across to the reprographics, where the reprographics will reply back with the required answer needed by sending a message back to that user.

For both the reprographics and the user the messages should be added to a stack and once the message is viewed it has to be removed from the 'new message' stack and added to the 'past messages' stack where it can later be viewed if necessary.

Credit remaining page

This page should output to the user how many print credits they have left to the end of the month, it will be reduced each time a print job form for that department is set. Currently each department has a limit of 1000 copies a month. The End user has requested to have an option to change this figure.

View Print Job forms

This page will allow the user to view all the jobs they have submitted to date. Each user will only be able to see all the jobs that they have just submitted and no one else. The only person that will be available to see all jobs from all departments is the reprographics. When a print job form is submitted, it is added to a stack which can later be viewed by the reprographics and also if the job is related to the user that has currently logged on to the system.

Accounts page (Only available to the reprographics)

The accounts page has to do the following things:

- It should have the option to find a particular department in the list to see what they have printed over a period of time; the reprographics will have the option to choose the period of time they want to see.
  For example, if they want to see what the Mathematics department have print over the past week, the reprographics will click on a button that will open a new window where it will ask which department and period you want it from, for this example you would select Mathematics and the period of the past week (e.g. 20th November 2009-25th November2009). After clicking ok, the system will output to the user a list of jobs the Mathematics department has done over the period of time selected.

- The accounts page iscurrently is split into three sections and this what the end users has requested the layout of the accounts page should look like on the new system:
  1. The main sheet that shows the overall cost for each department and the different categories the costs for each department fall into, for example costs for specials or just normal copies done.
  2. Specials – This page is used to calculate the total cost if the user department had chosen to have any special features, it includes a mini calculator so that if the reprographics wanted to change the cost of an item in the special features, it can quickly calculate how much it will cost by using the calculator. After observing the calculator and speaking with the user, I found out that the calculator has to at least do the basic operations (e.g. addition, subtraction, multiplication and division).
  3. Settings - this opens up the cost file and it enables the user to change the any of the costs, for example if they want to increase the price of printing a normal black and white page to 20p, they would just change the number. For the new system the user has requested to have a separate function that deals with this so that all the user end user has to do to change a price, is select from the list the item that they want to change and then input the new price. They should then have option to click a button that changes the price automatically by opening the list price and updating the price list.

- The main section of the accounts page should have the option of sorting the list in order of department, number of copies, date job submitted or userid.

The accounts page will only will available on the reprographics account only.

Objectives for the proposed system

General objective

To create a computer system for printing and storing print job requests from different users, including having an internal email system and also having features so that certain data can be extracted from the system to be viewed and interpreted.

Specific objective

1. To create a system so that the print job request is all done on computer
2. Each department has there own username and password
3. Each department is set a limit to how much of colour and black & white they can print
4. When they fill out the print job form online, all the cost and finance are calculated automatically and sent to the administrator (reprographics).
5. The computer decides if the print job is allowed or not.
6. If allowed it is sent to the required print queue for it to be printed
7. If not, a message an error message is outputted to the user giving reasons to why it has not been accepted.
8. Create a form that displays all the accounts and cost so far
9. Create a form that displays how much each department has printed separated into colour and black & white
10. Option to logoff
11. To create a user menu so that when the user logs in to the system they will see the main menu screen.
12. Option to close the window
13. Option to click for help to see what the required fields on the form mean
14. To create a main menu for the reprographics so when the reprographics logs on they will see the reprographics main menu.
15. To have an option so that all required fields are filled in the correct format before the form is sent to be processed.
16. To have the school logo on the username & password screen
17. To have the option on the 'print job form' to select the date you would like the job to be completed by.
18. Give a message when then print job is over 700 pages saying that this job will not be done in school as it is more efficient to do outside
19. To create a record that shows the accounts of the total cost of each print job that takes from the departments, place and this only be shown to the administrator(End user)
20. To set a limit on the number of black & white copies they are allowed to do each year which is currently 1000
21. To have a the option of generating a random security number which needs typed in before a print job form is to be submitted
22. To have a the option of generating a random security number which needs typed in before a send message form is to be submitted
23. To have the option to view new and past messages both for the user and reprographics.
24. To have the option for the reprographics to add a new user
25. To have a mini ca;culator which can do simple sums such as add, subtract, divude and multiply
26. Once a new user is created, a print credit profile is automatically created

27. To have the option to view a selected message
28. To have the option to see how many credits you have left
29. To give the option for the reprographics to change or add print credits for a user
30. To allow the user to view their past print job forms
31. To allow the reprographics to view a selected job when shown in the print job queue screen
32. To allow the reprographics to sort the accouts in UserID, requested date or total cost
33. To allow the user to pick a certain specific period of when print job forms have been submitted.
34. To have the option of view a selected past print job form
35. To have the option to send messages to the reprographics and for the reprographics to send messages back to the user
36. To allow the reprographics to change one or all the special price items
37. To give the option to the reprographics to view the prices of all the items

Prospective users

Other than the reprographics, each teaching departments in the school will need to have access to the system. There are about 30 departments so the new system will need to be able accommodate for this amount of users.

User needs and acceptable limitations

Hardware constraints

There are three different printers:

1. Colour printer
2. Black and white printer that also punches holes and staples for a limited number of papers
3. Black and white printer that puts holes for binding also punches holes and staples a lot more papers at once than the other printer. This printer is usually used for big jobs as it can handle with more papers all at once.

There are two computers, one which is mainly used by the reprographics to enter details onto the spreadsheet and to manage all the equipment in the room including dealing with the print jobs. The other computer is used mainly by other members of staff to do jobs such as uploading their work to show to the reprographics what they would like to be printed.

There are also other machines that are processed manually by hand such as a paper cutter, a binder, a laminator and one that folds paper into a booklet.

How the data will be stored?

The data will be stored in an XML file. There will be 7 XML files, each for used for different sections of the system.

1. To store the reprographics messages
2. To store details of the cost of each job submitted
3. To store details about each of the users
4. To store details of the printer credits for each user
5. To store details about each print job form that is submitted

6. To store message sent by the users
7. To store the cost of the special prices

Data structure and data types

Print job form XML file

Variable name	Data type
Job number	Integer
User ID	Integer
Staff initials	String
Request date	String
Copy supplied as	String
Print type	String
Paper type	String
Paper size	String
Number of copies	Integer
Paper colour	String
Back to back	Boolean
Collated	Boolean
Stapled	Boolean
Wiro-binding	boolean
Laminate	Boolean
Glue bound	Boolean
Transparencies	Boolean
Two-hole punched	Boolean
Folded	Boolean
Proof required	Boolean

Account details XML file

Variable name	Data type
Job number	Integer
User ID	Integer
Staff initials	String
Request date	String
Number of copies	Integer
Total cost	Integer

User details XML file

Variable name	Data type
User ID	Integer
Department	String
Password	String
Confirm password	String
Security question	String

Security answer	String

Messages XML file

Variable name	Data type
Message number	Integer
User ID	Integer
Name	String
Date submitted	String
Topic	String
Message	String

Reprographics sent messages XML file

Variable name	Data type
Message number	Integer
userID	Integer
Name	String
Date sent	String
Topic	String
Message	String

Special prices XML file

Variable name	Data type
A3	String
A4	String
A5	String
Normal	String
Card	String
Blackandwhite	String
Colour	String
Papercolour	String
Paperbandw	String
backtoback	String
Collated	String
Stapled	String
Wirobinding	String
Laminate a4	String
Laminate a3	String
Gluebound	String
Transparencies	String
Twoholepunched	String
folded	String

Print credits XML file

Variable name	Data type
UserID	Integer
Printcredits	Integer

Each string data type uses 255 charaters (memory allocated is as required) and integer data types uses a signed 32-bit integer (memory allocated is 4 bytes)

Software constraints

The school uses windows XP which is perfectly fine with what the reprographics does on it and also this will be suitable to run the new system on.

Time constraints

After speaking to the end-user, they would like to have the new system ready by the end of March 2010.

User's knowledge of information technology

The user has sufficient knowledge of ICT to be able to work with this new system.

Data flow diagrams (existing and proposed system) to level 1 (Shown on next page)

Considerations of alternative solutions

1. Using Microsoft access
   The advantages of using Microsoft access were that you can have a fully functional, relational database that can be built in a few minutes and the user interface elements are quite powerful.

   However there are some disadvantages with it such as access can support a maximum of 255 concurrent users and therefore is not a very good program to have for running an enterprise-level data storage solution. After doing some further research on Microsoft access I found out that Microsoft Access supports a maximum database size of 2GB.

2. Using Microsoft word
   The advantages of using Microsoft word is that it is widely used so a lot of people will be familiar with using it and it supports a number of file types.

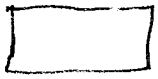
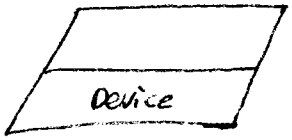
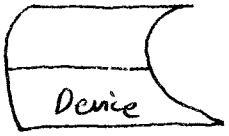
   The disadvantages of Microsoft Word is that it can crash which can be quite annoying when you are in the middle of for example, completing the print job for.

3. Doing the whole project in Visual basic 2008
   Microsoft Visual basic has a very clear and understandable structure. The language is very simple, particularly as to the executable code. Visual Basic provides a comprehensive interactive and context-sensitive online help system which can be very helpful in times when you want to know what the syntax for a particular function is or to see how you can do certain things such as writing classes.It does come with disadvantages and that is that Visual Basic is a proprietary programming language written by Microsoft, so programs written in Visual Basic

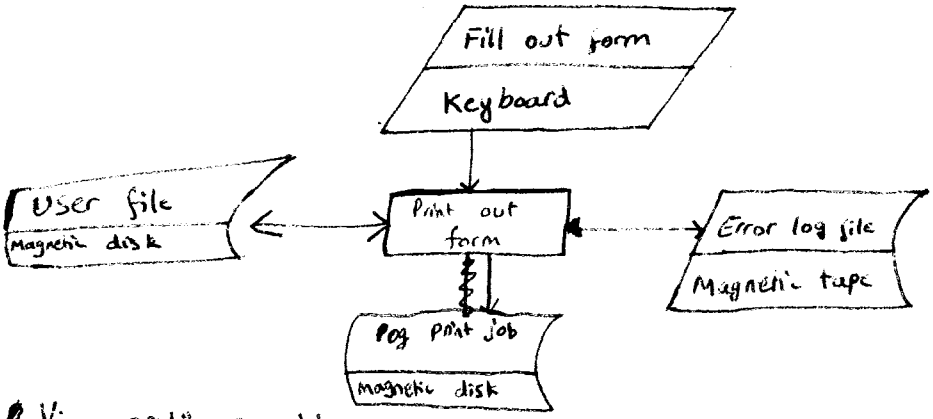
cannot easily be transferred to other operating systems.

4. Using Visual basic joint with Microsoft access

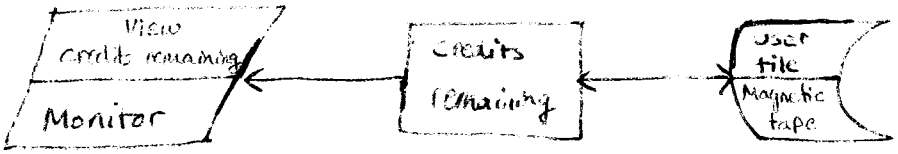
System Flowchart

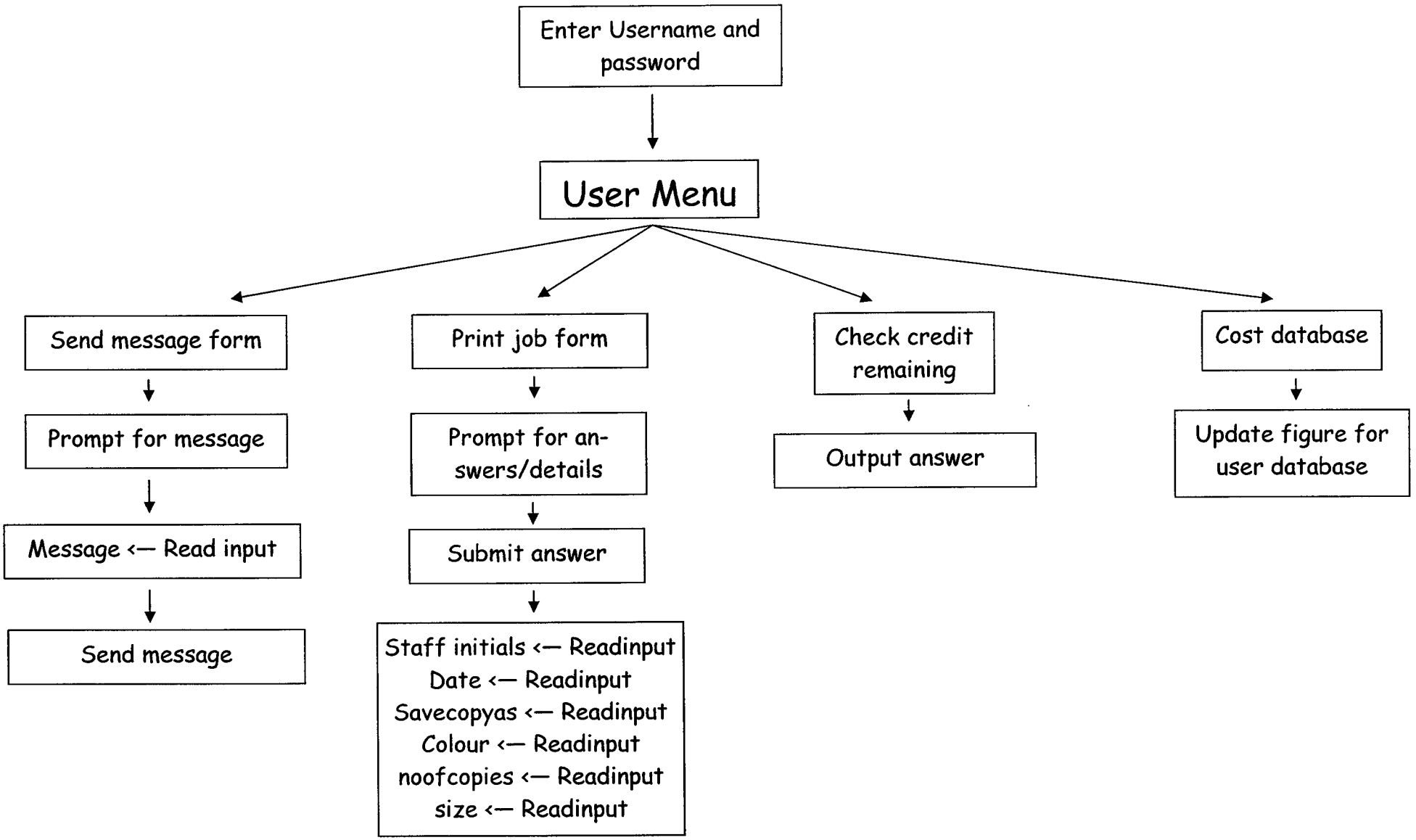
Process	Input/output	Online storage	Flow line
			
This could be a programme or a manual process	This could be data keyed in at a keyboard, a bar code read wit a bar code reader	This could be a database on magnetic disk, a data file on CD-RW	Links the symbols to define sequence and direction of flow

Print out form



View credits remaining




Enter Username and password
User Menu
Send message form
Prompt for message
Message <— Read input
Send message
Print job form
Prompt for answers/details
Submit answer
Staff initials <— Readinput
Date <— Readinput
Savecopyas <— Readinput
Colour <— Readinput
noofcopies <— Readinput
size <— Readinput
Check credit remaining
Output answer
Cost database
Update figure for user database

Justification of chosen solution

After careful consideration on the alternative forms in solving the problem I have decided to do the whole project in Visual Basic. A reason for this is that it has an ease of learning, the syntax is simpler than other languages. The visual environment is very friendly and also it is widely used, therefore, well understood.

Microsoft Visual Basic (VB) also ticks all the boxes of the objectives that I will have to perform. VB gives the option of designing the screens, writing the code to link them together and also write and store files.

Microsoft word will not be suitable for what I am doing as the design of the program can be quite difficult as it does not have the entire feature such as data validation on things like the date format. Also there is the cost, Microsoft word can cost around £50-60 which would be expensive there were a lot of user that needed it on their machines to run the program. Another us that it has a lot of extra features that most people won't use and that can sometimes get in the way of getting things done.

This means that using Microsoft Access is not a feasible solution to solve the problem. Reasons for this are that, one of the objectives that was stated by the end-user is that in the future the program should be capable of having more than 1000 users so that everyone include pupils had a username and password so Microsoft Access would not be suitable as it can only have 255 concurrent users.

Section 2

Design

Overall system design

If there is any problem with anything like for example, if the print job form has not been sent, clearly instructed error messages will be outputted giving information to the user on what to do in order to solve the problem.

All text will be in Consolas, this is because this type of font is easy to read making it is for the user or reprographics to understand what the writing is saying.

The design of each screen will be simple so that it is easy to see what is happening.

The 'new messsage' and 'past message' screens will be different for the reprographics and the user. For the reprographics, all messages that are sent to user will be saved to a separate XML file. This wll mean that for system maintenance it will be easier to track down any errors or there needs to be a changes needed to be done to the code as the person maintaining the system will know for example if the problem is from the user end or the reprographics.

Classes that are going to be created for the system and what the purpose of each one is:

User sent messages

- Opens messages.xml file
- Locates child node (number)
- Stores information collected from the XML file in a variable
- Also used to check the number of entries in the XML file

This will class will open the XML file that contains the messages that are sent by the users of the system. The data extracted from the XML file will be used to add the messages to the new message stack and past message stack when the reprographics wants to view either new or past messages which have been sent by the users.

It will also save the contents of the messages to the XML file.

Userprofile

- Opens userdetails.xml file
- Locates child node (number)
- Stores information collected from the XML file in a variable
- Also used to check the number of entries in the XML file

A function to save the user details to the XML file.
In addition when the reprographics fills in the form to add a new user, this class will be used to save the information to the XML file.

Accounts

- Opens accounts.xml file
- Locates child node (number)

- Stores information collected from the XML file in a variable
- Also used to check the number of entries in the XML file

Information gained will be used to add the account information on to the stack so that it can be viewed by the reprographics when they are viewing the accounts page. The class will also be used to save the financial information generated from every print job form submitted. It will also be used to sort the account details in UserID, date submitted or total cost and the extract the relevant data required by the reprographics if they want to view certain information of the accounts. For example if they want to see what the Economics department has printed in the past week.

Reprographics messages

- Opens reprographicsmessages.xml file
- Locates child node (number)
- Stores information collected from the XML file in a variable
- Also used to check the number of entries in the XML file
- Save messages sent by the reprographics to the XML file

This will class will open the XML file that contains the messages that are sent by the reprographics. The data extracted from the XML file will be used to add the messages to the new message stack and past message stack when the reprographics wants to view either new or past messages.

The class will be used to save the messages sent via the reprographics to the XML file.

Printer credits

- Opens printcredits.xml file
- Locates child node (number)
- Stores information collected from the XML file in a variable
- Also used to check the number of entries in the XML file

The data extracted from the XML file will be used to check to see if the user has enough credits in order to allow the print job form that has been submitted to proceed. Every user is given 1000 printer credits every month and every print job form they submit, their printer credits is updated (For every copy the user prints, one printer credit will be deducted).

The 'printer credits' class will also have a function to update the printer credits for any user which will be stated by the reprographics.

Every user that is created, the 'printer credits' class will be used to generated a profile for that user and will be saved to the XML file via the class.

Special price list

- Opens specialprices.xml file
- Locates child node (number)
- Stores information collected from the XML file in a variable

- Also used to check the number of entries in the XML file

This data will be used to display the prices of the special items and also to change the prices if necessary. The data will also be used to calculate the total cost of a print job form using the information gained from the XML file.

Print job form

The purpose of this class is to save the information on the print job form filled out by either the reprographics or users to 'Print job form' XML file.
It will also be used to open the file and extract certain information which is used for example when the reprographics is viewing the print jobs that need to be done on the 'Print job queue'.

Also when the job is removed from the print job queue the class is used to set the to set the print job form status to 'true' which mean that the job has been completed.

Queue class
The queue class will be used to add the print job form onto the the queue so when the reprographics wants to view the jobs that need to be done, they will be able to do so. The class will do the following things:

- Initialise the queue
- Add print job forms to the queue
- Sort the queue in date submitted
- Delete print job forms from the queue (this is when the job has been completed)
- Show the contents of the queue

A similar function in the class that is used to add the print job form to the print job queue will be used to add messages to the new and past message stack, both for the reprographics and the users, add the accounts information to the accounts stack on the 'accounts' page.

Below shows a table of the classes used and the purpose of each one.

Class: Print job form

Function/procedure name	Purpose of function/procedure	Variables used in function/procedure	Purpose of variables
Job complete	Get the job which is at the top of the print job queue and changes its status on the XML file as job completed	Job number (integer)	Opens XML file 'printjobform.xml' Finds the job node number on the XML file that needs to be changes using the class 'find job' Stores the job number node that was found from the find job class
add	Saves the print job form details to the XML file 'print jobform.xml'	Print job form xml	Opens XML file 'printjobform.xml'
length	See how many print job forms have been submitted	Printjob Printjobformitems	Opens XML file a list of printjobforms which are a child of the job node

The following functions will each get the information (stated by their name) from the XML file depending on which child node number is required:

So for example, child number 2 would select the third print job form (as XML files start from 0) and from that selects what the information that would be required (e.g. staff initial).

Job number
User ID
Staff initials
Request date
Copy supplied as
Paper type
Paper size
Number of copies
Paper colour
Back to back
Collated
Stapled
Wirobinding

Laminate
Glue bound
Transparencies
Two hole punched
Folded

Class: Special price list

Function/procedure name	Purpose of function/procedure	Variables used in function/procedure	Purpose of variables
Change special price list	Gets the selected item that the reprographics to the change the price to and updates the price using what the reprographics used when they fill in the form	Special prices	Opens XML file 'special prices.xml'
add	Saves the special price details to the XML file 'specialprices.xml'	Special price xml	Opens XML file 'specialprices.xml'
Calculate total price of print job	This function calculates the total cost of the job using what the user has just submitted from the print job form.	Totalcost (string)	Stores the total cost

The following functions will each get the information (stated by their name) from the XML file depending on which child node number is required:

Papersizea3
Papersizea4
Papersizea5
Normal
Card
Paper colour
Bwprint
Colour print
Plain paper
Backtoback
Collated
Stapled
Wirobinding

Laminatea4
Laminatea3
Gluebound
Transparencies
Twoholepunched
folded

The following procedures change the price of the special item and replace the initial price in the XML file 'Specialprices.xml':

A3
A4
A5
Normal
Card
Balackandwhite
Colour
Paper colour
Paper bandw
Backtoback
Collated
Stapled
Wirobinding
Laminatea4
Laminatea3
Gluebound
Transparencies
Twoholepunched
folded

Class: Print credits

Function/procedure name	Purpose of function/procedure	Variables used in function/procedure	Purpose of variables
Saveprintcredits	To collect the required details from the print job form and add and save it to the Print job credits XML file	n/a	n/a
Update print credits	Opens the XML file 'printcredits.xml' and updates the print credits for that user that has just submitted a form. It also check to see if it has a enough	Current print credit number	Stores the new value of print credits

	credits to proceed with the print job form they have just		
Length	See how many print credit profiles there are	Message Print credit details	Opens XML file a list of print credit which are a child of the user print credit profile node

The following functions will each get the information (stated by their name) from the XML file depending on which child node number is required:

UserID
Printcredits

Class: user profile

Function/procedure name	Purpose of function/procedure	Variables used in function/procedure	Purpose of variables
Save user	Saves user details to the XML file 'userdetails.xml'	Usersxml	Opens XML file 'userdetails.xml'
length	See how many print credit profiles there are	Userdetails userdetail	Opens XML file a list of user which are a child of the users node

The following functions will each get the information (stated by their name) from the XML file depending on which child node number is required:

UserID
Department
Password
Securityquestion
Securityanswer

The 'change password' procedure is used to change the password of the user and replace this with the initial one in the XML file.

Class: User sent message

Function/procedure name	Purpose of function/procedure	Variables used in function/procedure	Purpose of variables
Message read	Set the me message status to read and saves this	Messagexml	Opens XML file 'messages.xml' Used to find the child node

	information in the XML file	Message number	number in the XML file 'messages.xml' of the message profile
Add message	Saves the message form details to the XML file 'print messages.xml'	Message details xml	Opens XML file 'messages.xml'
lengths	See how many messages there are in the XML file 'messages.xml'	Message Message details	Opens XML file 'reprographicssentmessages.xml' a list of message which are a child of the messages node

The following functions will each get the information (stated by their name) from the XML file depending on which child node number is required:

Message number
UserID
Name
Date submitted
Topic
Submittedmessage
Read

Class: Reprographics sent message

Function/procedure name	Purpose of function/procedure	Variables used in function/procedure	Purpose of variables
Set message to read	Set the me message status to read and saves this information in the XML file	Messagexml Message number	Opens XML file 'messages.xml' Used to find the child node number in the XML file 'messages.xml' of the message profile
Save reprographics sent message	Saves the message form details to the XML file 'print messages.xml'	Reprographics message sent details xml	Opens XML file 'messages.xml'
lengths	See how many messages there are in the XML file 'messages.xml'	Message Message details	Opens XML file 'reprographicssentmessages.xml' a list of message which are a child of the reprographics sent messages node

The following functions will each get the information (stated by their name) from the XML file depending on which child node number is required:

Message number
UserID
Name

Date submitted
Topic
Submittedmessage
Read

Class: accounts

Function/procedure name	Purpose of function/procedure	Variables used in function/procedure	Purpose of variables
Save account details	Saves the accouts details to the XML file 'accountdetails.xml'	User account details xml	Opens XML file 'accountdetails.xml'
Length	See how many messages there are in the XML file 'messages.xml'	Message Message details	Opens XML file 'accountdetails.xml' a list of user account details which are a child of the accounts node

The following functions each get the information (stated by their name) from the XML file depending on which child node number is required:

UserID
Staff initials
Department
Date submitted
Number of copies
Total price

Class: Queue

The 'add form to queue', initialises the print job queue, clears the queue items, adds items to the queue, removes items from the queue, sorts the contents of the queue and outputs the queue contents.

Each of the procedures below will, initial the stack, clear the stack, add items to the stack, remove items from the stack and output the stack contents:

- Add account details
- Add new message to stack
- Add past message to queue
- Add to list past jobs
- Add user past message to stack
- Add user new message to stack

The procedures below store the information collected from the 'send message', 'send message (reprographics)', 'login screen', 'print credit', 'print job form' and 'special price items' screens and stores them in variables where they are used for example to save the information to the XML file. The procedures use 'property get':

- Get change price details (Gets details from the 'change price details' form and stores them in the variables so that they can be used by the other class to perform the rest of the procedure)
- Get user details (Gets details from the 'add new user' form and stores them in the variables so that they can be used by the other class to perform the rest of the procedure (uses property get function))
- Get details for print credit change (Gets details from the 'change print credits' form and stores them in the variables so that they can be used by the other class to perform the rest of the procedure (uses property get function))
- Get entered login details (Gets details from the 'login screen' form and stores them in the variables so that they can be used by the other class to perform the rest of the procedure (uses property get function))
- Get message details (Gets details from the 'send message' form and stores them in the variables so that they can be used by the other classes to perform the rest of the procedure (uses property get function))
- Get new prices on everything (Gets details from the 'change all special item prices' form and stores them in the variables so that they can be used by the other classes to perform the rest of the procedure (uses property get function))
- Get security answer (Gets details from the 'change all special item prices' form and stores them in the variables so that they can be used by the other classes to perform the rest of the procedure (uses property get function))

The function 'does user have a new message' is uses the class 'user sent messages' and ' reprographics sent messages' to see if there are any messages in the XML file that has not been read.

The 'find user' and 'find job' functions will use the class 'userprofile' and 'print job form' to extract the required information and see if for example the login details mathch what is stored in the XML files or if for example to find the job profile when the reprographics wants to view the details of the print form job.

Algorithms

Print job form

'Submit' button pressed
Checkcredits(userID)
Calculate print credits needed to do process this job
If user has enough credits then
    Gather details of form
    Calculate job cost
    Update user print credits(userID)

Save form
Add form to reprographics print job queue
Output message 'Job submitted succesfully'
Exit print job form screen
Show Main menu
Else
Output message 'Your job was not sent, please check the form and try again'
End if

Login screen

'Login' button pressed
Check username and password
If correct then
If username = reprographics then
Show reprographics main menu
Else
Show user main menu
End if
Elseif not correct then
Output message 'Invalid username/password entered, please try again'
End if

User menu

If 'Print job form' button pressed then
Show print job form screen
Elseif 'Send message' buton pressed then
Show send message screen
Elseif 'View print job forms' button pressed then
Show view print job form screen
Elseif 'View messages' button pressed then
Show view messages screen
End if

Send message

'Send' button pressed
Check to if all details have been filled in
If ok then
Gather details
Save message to file
Add message to reprographics message stack
Close 'send message' screen
Show user menu
Elseif not ok then
Output message 'Message not sent, please fill in all the boxes'
Endif

Forgotten password 1st stage

Check userid entered
If valid then
    Output message 'UserID verified'
    Exit 'forgotten password 1st stage' screen
    Show 'forgotten password second stage' screen
Elseif invalid then
    Output message 'Invalid UserID entered, either try again or see the reprographics'
    Exit 'forgotten password 1st stage' screen
    Show login screen
End if

Forgotten password 2nd stage

Check to see if security question and security answer match
If they do match then
    Output message 'Your password is: (*Show password*)'
    Exit 'forgotten password 2nd stage
    Show login screen
Else if they don't match then
    Output message 'Invalid security answer entered, either try again or see the reprographics'
    Exit 'forgotten password 2nd stage
    Show login screen
Endif

Change add print credits

Find userid profile number (userID entered)
If userID is a valid userID then
    Update their print credits by adding the amount specified
Else
    Output message 'Invalid userID, please try again'
End If

Add a user

Check to see all required fields are added
If everything is filled then
    Gather details
    Save to user XML file
    Exit 'add a user' screen
    Show user menu
Else
    Output message 'Please fill in all required areas'
End if

Accounts Sort

Find number of child nodes in 'accounts' XML

```
If sort selected = "date submitted" then
        For number = 1 to number of child nodes in 'accounts' XML
        Array(number)= add childnode (number-1)
        Next
        Do
                No more swaps =true
                For number = 1 to number of child nodes in 'accounts' XML
                If array(number) > array(number+1) then
                        No more swaps =false
                        Tempory value = array(number)
                        Array(number) = array(number + 1)
                        Array(number + 1) = Tempory value
                End if
        Loop until no more swaps = true
        Add array() contents to stack
ElseIf sort selected = "number of copies" then
        For number = 1 to number of child nodes in 'accounts' XML
                Array(number)= add childnode (number-1)
        Next
        Do
                No more swaps =true
                For number = 1 to number of child nodes in 'accounts' XML
                If array(number) > array(number+1) then
                        No more swaps =false
                        Tempory value = array(number)
                        Array(number) = array(number + 1)
                        Array(number + 1) = Tempory value
                End if
        Loop until no more swaps = true
        Add array() contents to stack
Elseif sort selected = "UserID" then
        For number = 1 to number of child nodes in 'accounts' XML
        Array(number)= add childnode (number-1)
        Next
        Do
                No more swaps =true
                For number = 1 to number of child nodes in 'accounts' XML
                If array(number) > array(number+1) then
                        No more swaps =false
                        Tempory value = array(number)
                        Array(number) = array(number + 1)
```

```
                    Array(number + 1) = Tempory value
                End if
        Loop until no more swaps = true
        Add array() contents to stack
Elseif sort selected = "Total cost" then
        For number = 1 to number of child nodes in 'accounts' XML
        Array(number)= add childnode (number-1)
        Next
        Do
                No more swaps =true
                For number = 1 to number of child nodes in 'accounts' XML
                If array(number) > array(number+1) then
                        No more swaps =false
                        Tempory value = array(number)
                        Array(number) = array(number + 1)
                        Array(number + 1) = Tempory value
                End if
        Loop until no more swaps = true
        Add array() contents to stack
Endif
```

Print job queue

```
Open 'print job form' XML file

Update
If job has not been completed then
        Add to job to queue
Else
        Ignore job
        Number = number + 1
End if

Remove job at top
If queue is empty then
        Output message 'no jobs to process'
Elseif queue is not empty then
        remove job at top of queue
        increment all other jobs up by 1
        set job as completed
End if
```

View selected message

```
Open 'message' XML file
Number = 0
If message selected = message child node (number) then
        Output message on to form
```

Message found=true
Else
Number = number + 1
Message found = false
EndIf
Loop until message found = true or number = number of child nodes in XML file

Show special price list
Open 'special price' XML file
Output price of each item using XML file

New message
Open 'message' XML file
Number = 0
If message has not been read in child node (number) then
Add to new message stack
Else
Number = number + 1
End If
Once message has been read
Remove from new message stack
Add to past messages stack

Past messages
Open 'message' XML file
Number = 0
If message has been read in child node (number) then
Add to new message stack
Else
Number = number + 1
End If

Measures planned for validation checks

Print job form

- Make sure 'number of copies' does not exceed 700. If it does an error message should be outputted saying 'Number of copies should be less than 701
- Check to see if all required fields have been filled in
- Check to see if right security answer has been entered

Send message screen

- Check to see if everything has been filled in
- Check to see if right security de has been entered
- Check to see if a valid userID has ben submitted

Login screen

- Check to see if correct userID and password has been filled in
- Check to see if everything has been filled in, nothing is left blank

An error message will be outputted to the user if something has not been done correctly.

Measures planned for system security

In order to use the system you will have to log in using the details from when the user signed up to user to the system. This way unathorised people won't be able to use the system.

Other security checks that will take place are for example when they want to reveal their password, they will have to go through some security checks like first verifying their userID and also the answer to their security question. And if they are correct then the password will be shown.

Overall test strategy in relation to the problem being solved and tested

Proposed test plan

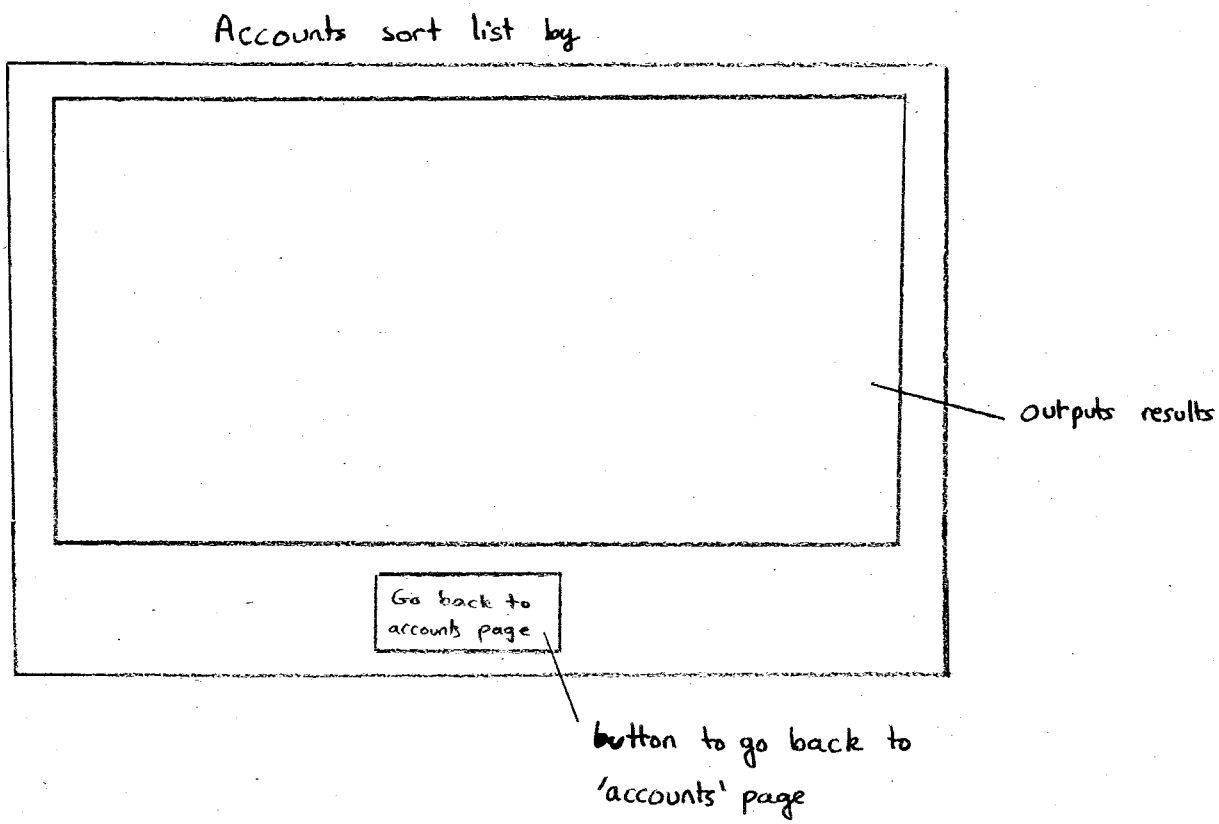
Test Series	Purpose of test
1	Test the flow of control: - Do the Menu choices (menu structure) - See if interface options go to the correct form or option - User can only choose-appropriate options
2	Validation of input data performed correctly.
3	Iterations and decisions performed correctly. Calculations and searches made in the account's page is outputted correctly. (White box testing and desk checking)
4	Data is saved into the correct array and files. (System testing)
5	The system produces the required results as per specification. (black box testing)

Detailed test plan

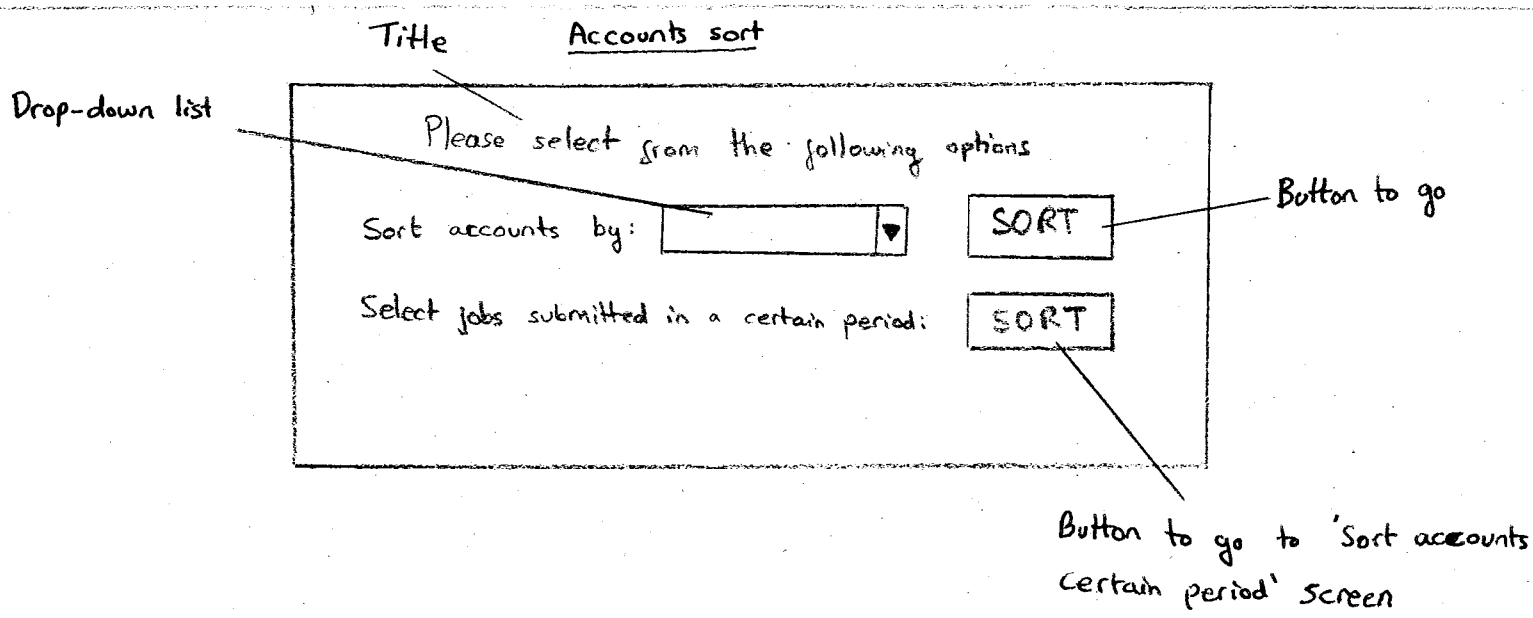
Test series and numbers	Purpose	Description	Test data	Expected result
1	Security and privacy	Validate to see that the password is hidden when typed and is replaced by '*'	Type some random letters (e.g. abcdefghi)	Letters should be replaced by a '*' symbol.
2	Validate number of copies allowed to be printed	Each department is allowed to print up to and including 700 copies per print job request, anymore should be declined by the system.	699 700 701	Accept value Accept value Decline value (Error message outputted)
3	Validate to see if user ID entered on 'Send message' form is a entered correctly and is checked by the system to confirm	If user enters a User ID that is incorrect, the system will confirm this by checking the list of valid user IDs and if the one entered is not there it should output a error message to the user and also an error message if nothing has been entered	2569 1111 2568 6589 'abs'	Accept Decline (Error message) Accept Decline (Error message) Decline (Error message)

4	Validate output message after print job form has been submitted	Once the user has finished completing the print job form and once they press the 'submit' button, the system should do some last minuet checks and if everything is fine and is sent to the reprographics, a message should be outputted to say that it has been sent. And if the form has not been sent, a message is output displaying reasons to why it has been declined	All field filled correctly Some field corrected correctly Nothing entered on form	Message output ('Your form has been successfully sent') Message output ('Your form has not been sent because the date has not been entered') Message output ('Nothing has been entered)
5	Validate sorting in accounts form	Check to see that data that is wanted is outputted to the reprographics so that if the reprographics wants to view all jobs submitted from Physics department for a certain period of time, the find button does this.	Physics jobs submitted last week Nothing entered Psychology jobs submitted yesterday Chemistry jobs submitted in a future date	All jobs printed by the Physics department is outputted Error Message output ('Nothing has been entered') Error Message output ('Invalid department entered') Error Message output ('Cannot show jobs of future event')

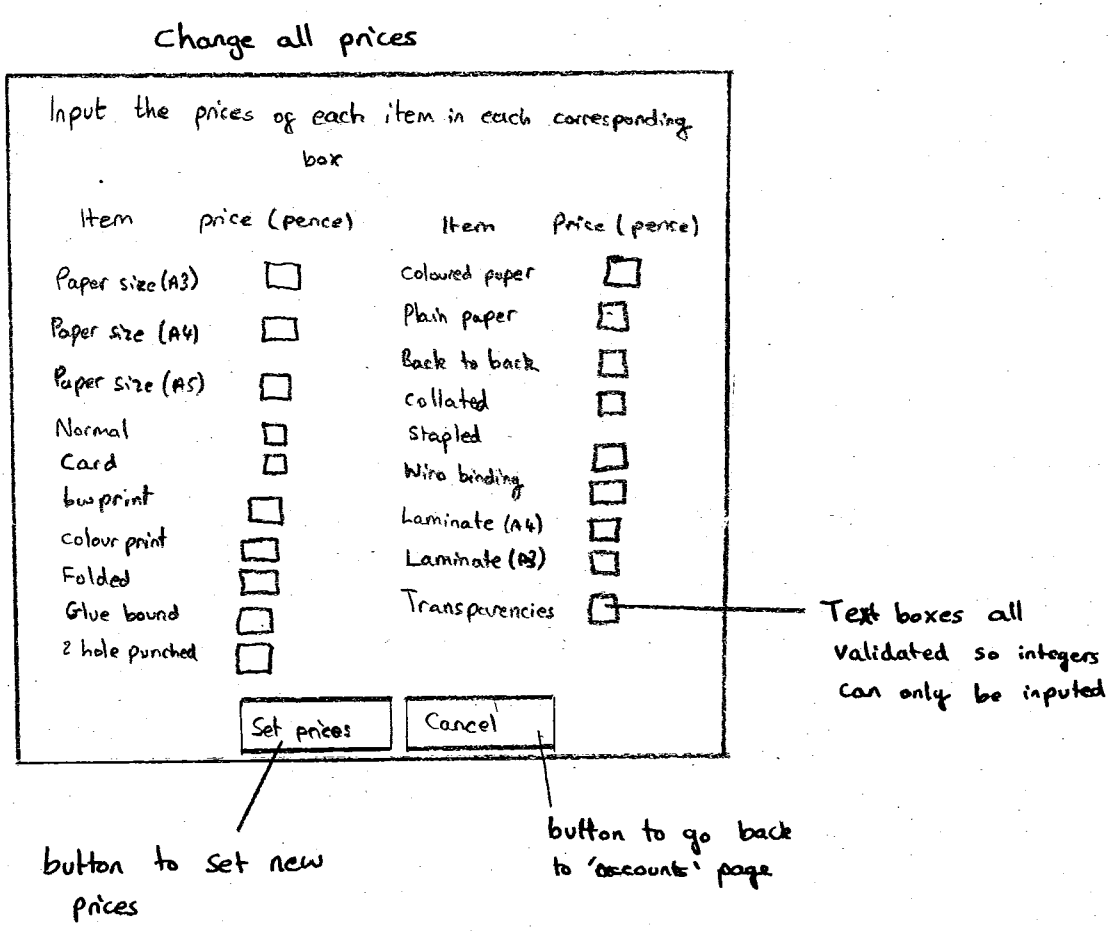
User interface design (shown on next page)



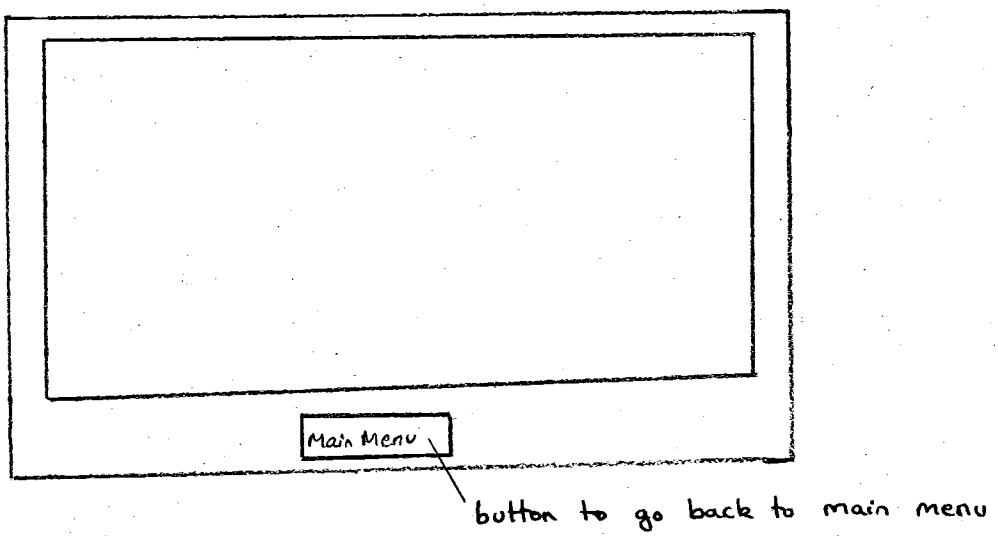
Depending on what you selected on the 'Accounts sort' window, the results will be added to this ~~screens~~ screen's stack



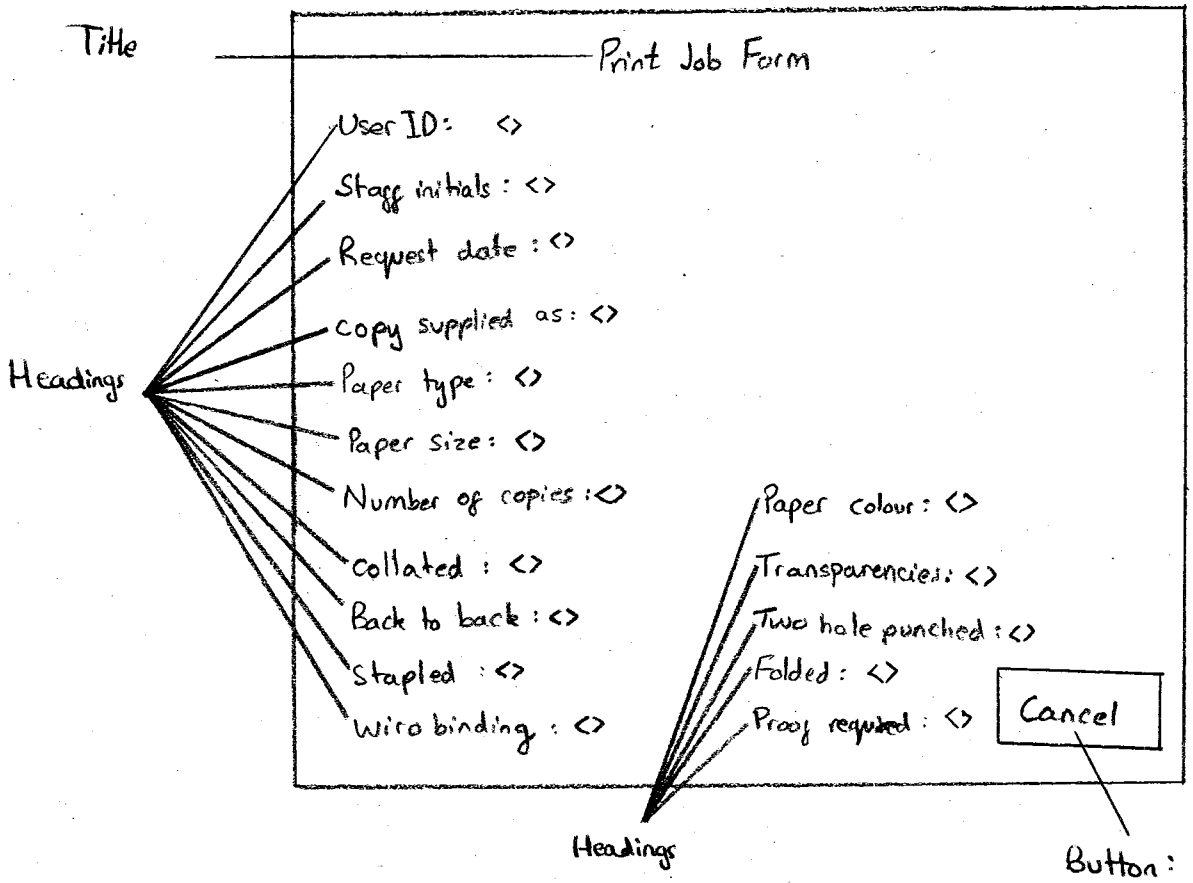
Drop-down list will have options: 1. User ID
2. Number of copies
3. Total cost
4. Date submitted



A message will be outputed to confirm new prices have be set. If there is an error, an error message will be outputted given clear instructions on how to solve the problem.

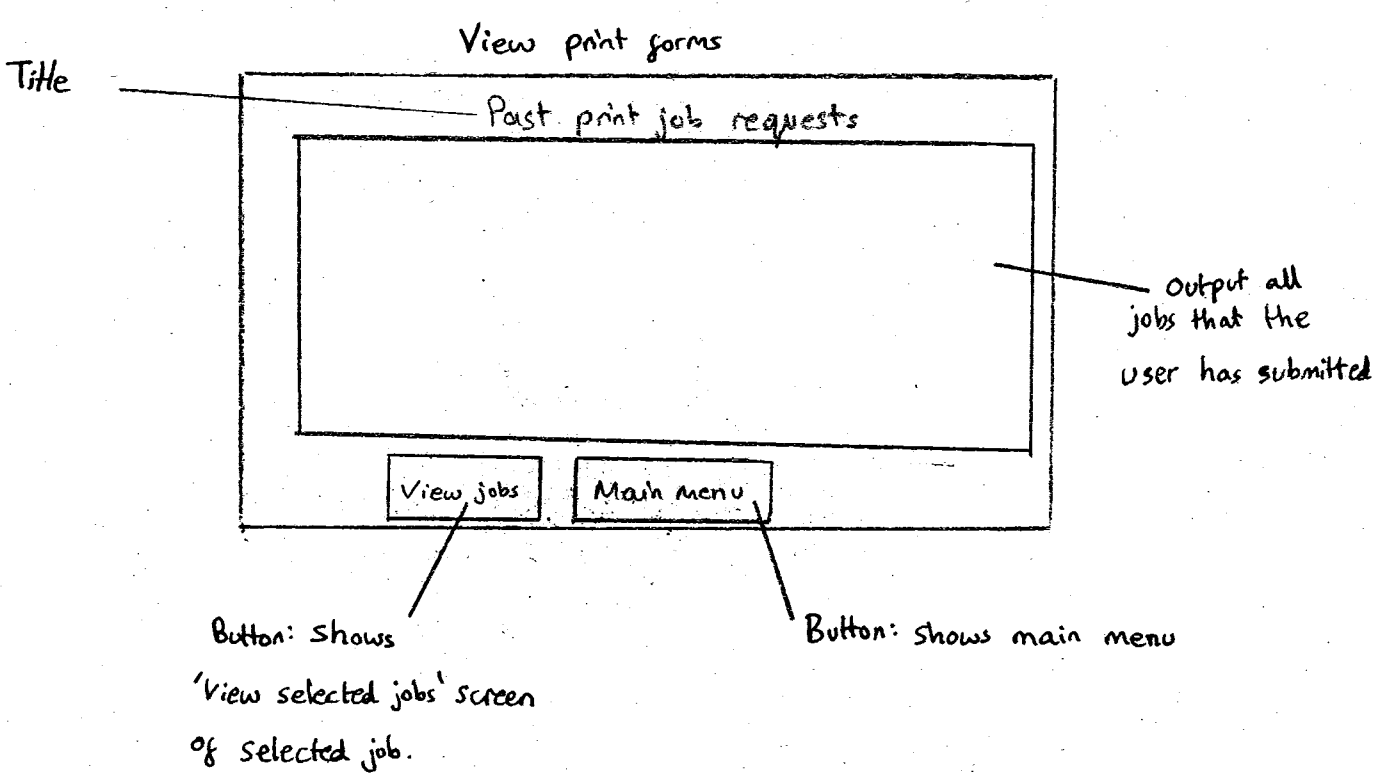


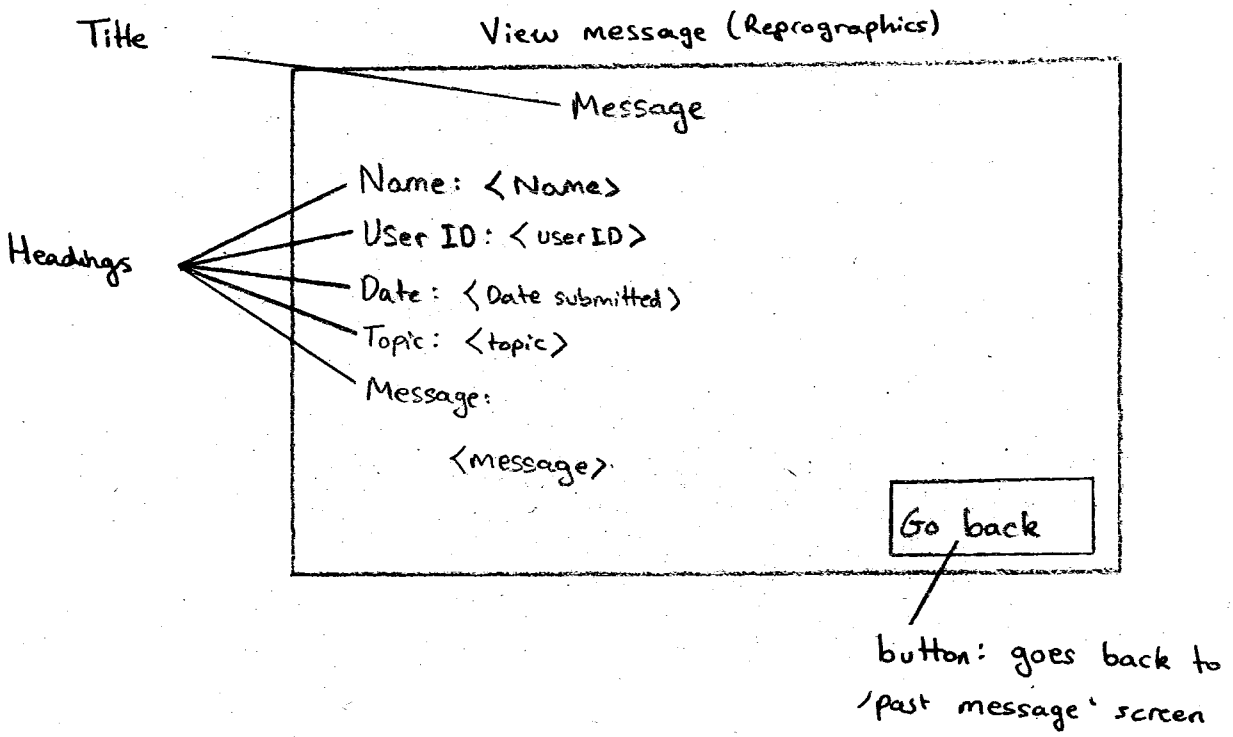
View print job forms



Button: goes back to 'main menu' for user and 'print job queue' screen for reprographics.

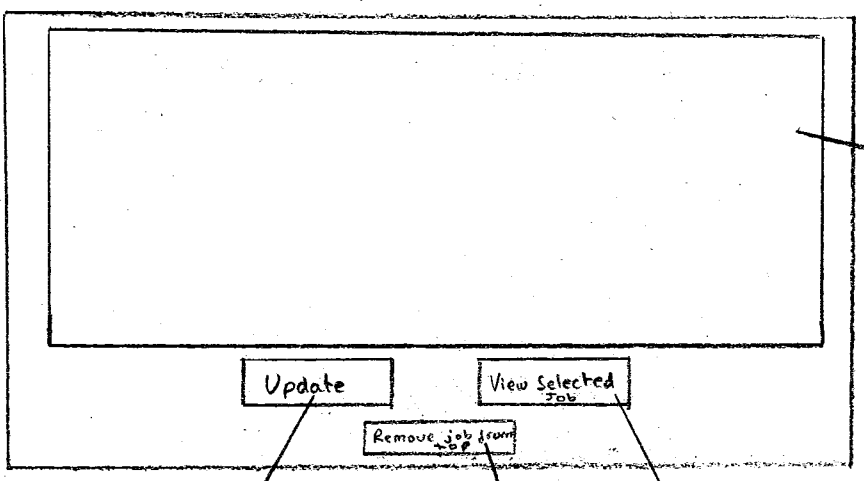
'<>' outputs details based on the XML file.





'< >' shows details of message which has come from the XML file.

Print Job Queue



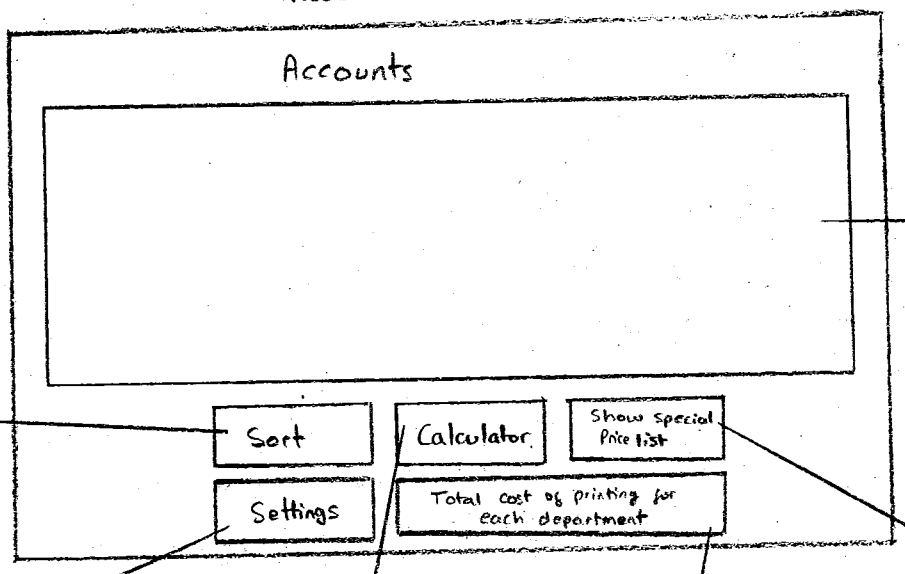
Print queue jobs

Button to update Print job queue

button to remove job at the top from the queue

buttons: Shows ~~Opens~~ details of Selected job

Accounts



Outputs accounts

Button: shows 'accounts sort' screen

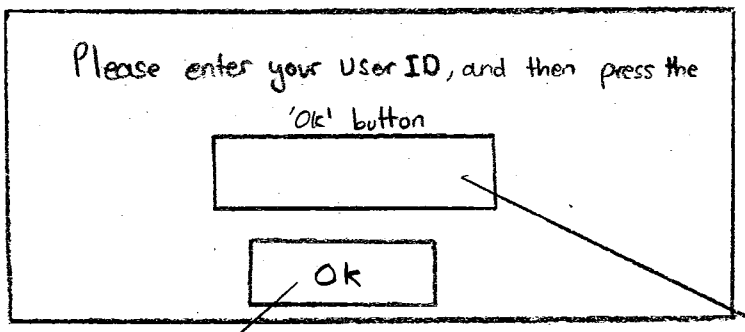
button shows 'Accounts settings' screen

button: shows 'Calculator' screen

button: shows 'total cost' screen

button: shows 'Show special price list' screen

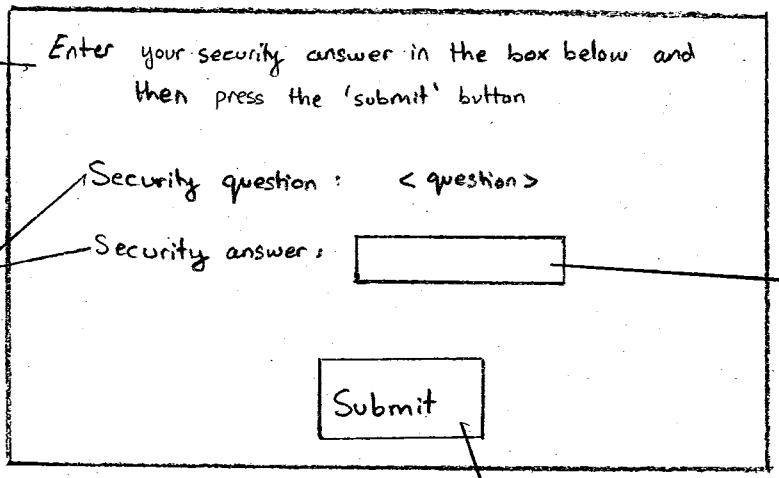
Forgotten password 1st stage



Validated text box so that integers can only be inputed

button to go to 'Forgotten password final stage' screen

Error message outputed if User ID not found.



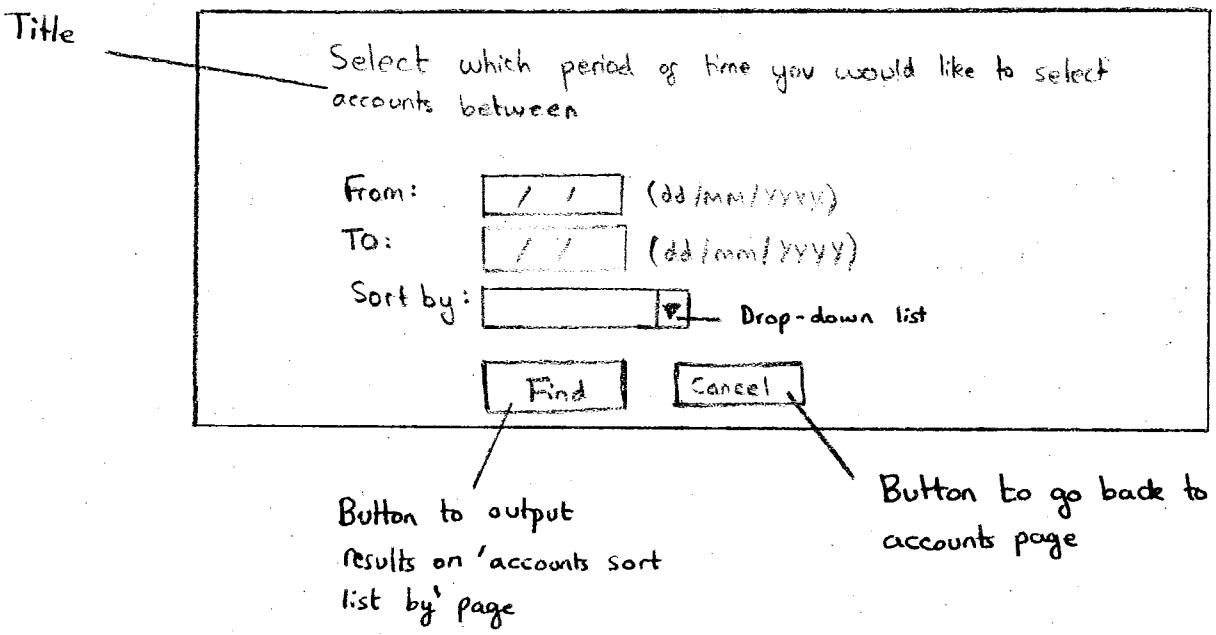
Title

Headings

Text box

button to output password if security answer is correct, if it isn't an error message will be outputted telling the use to either try again or see reprographics.

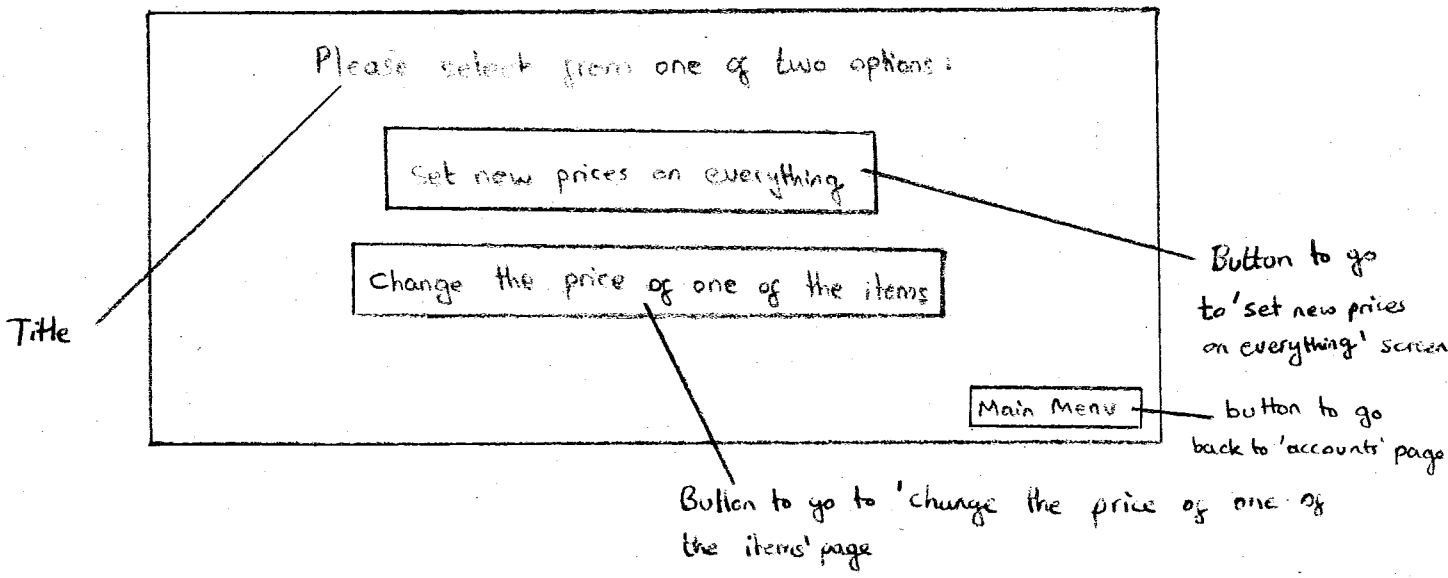
Accounts sort certain period



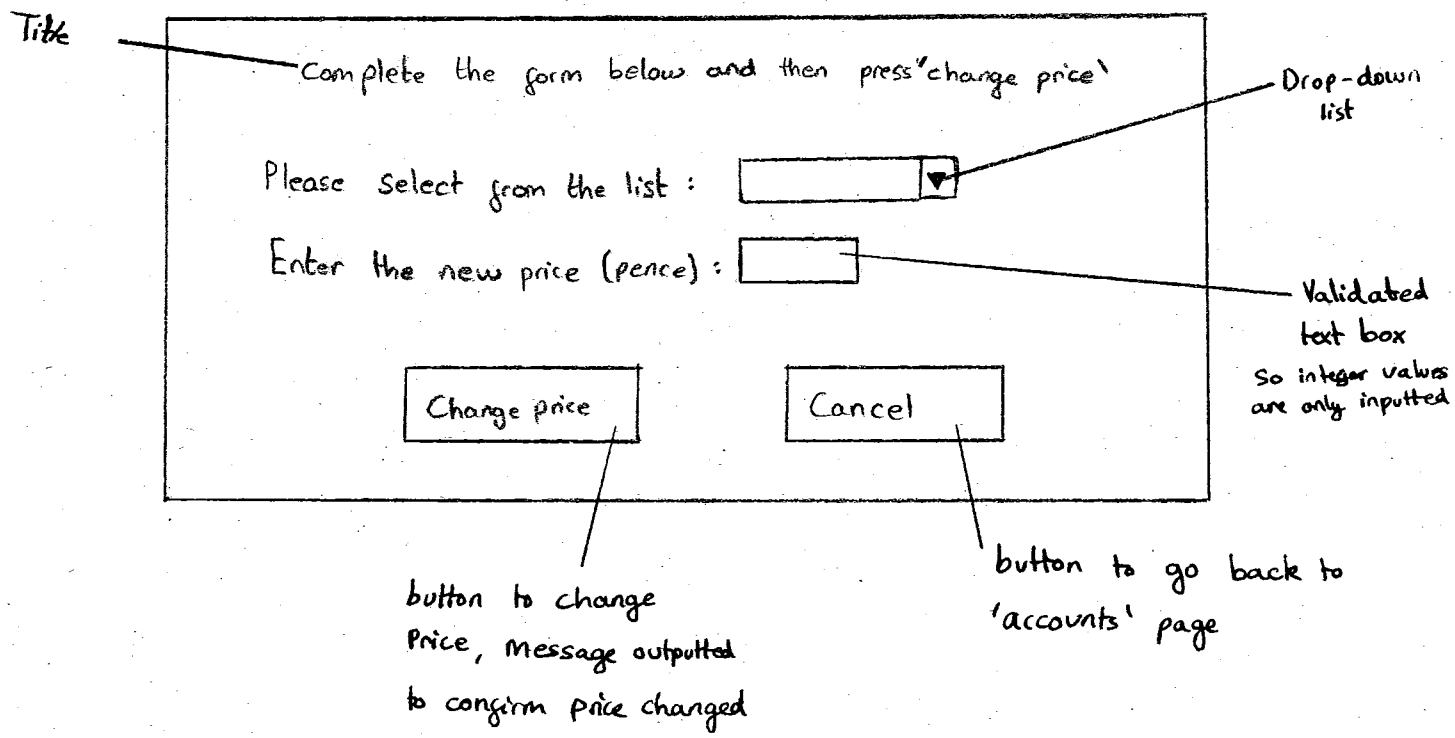
'From' and 'To' textboxes will be validated so date will be inputted in correct format. (dd/mm/yyyy).

Sort by; will be a drop down list box which the user can choose from four options:
1. User ID
2. Number of copies
3. Total cost
4. Date submitted

Accounts settings

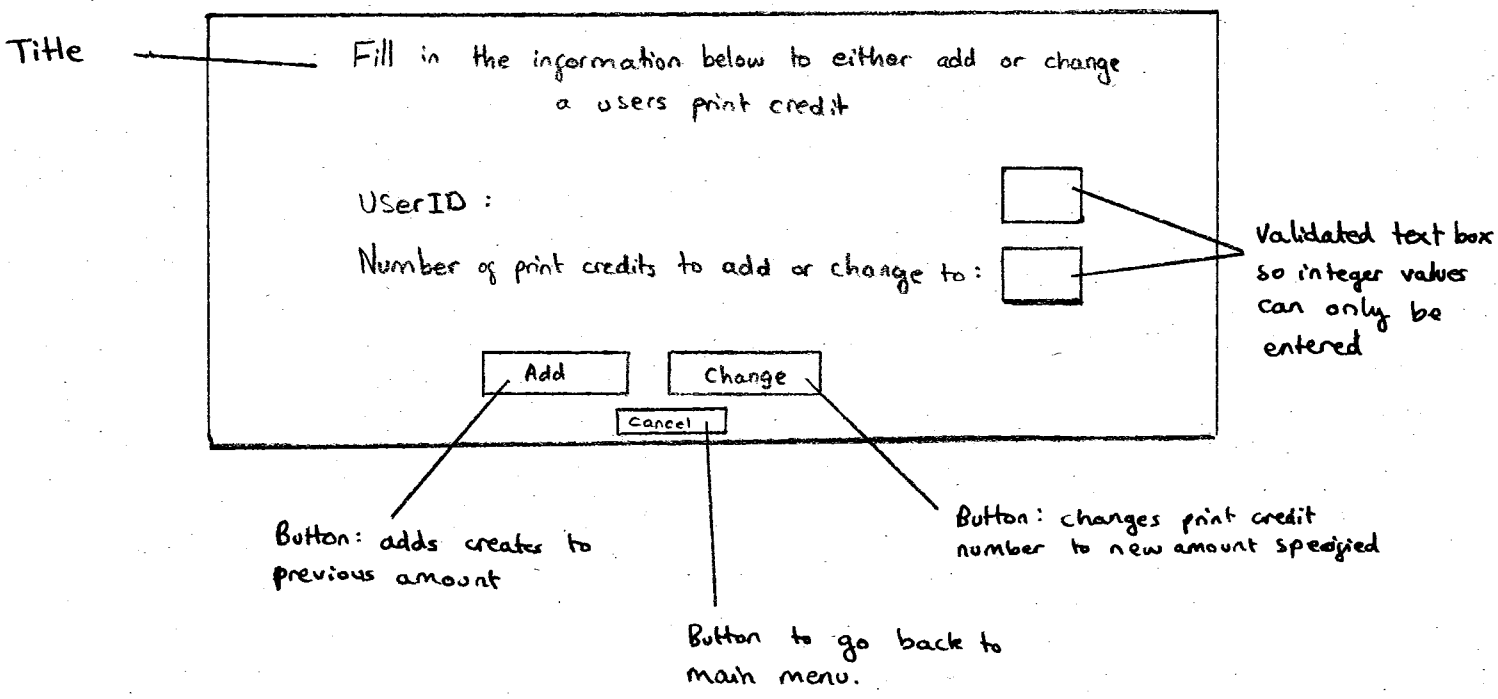


Change one special price item



Drop down list will have options :
- A3
- A4
- A5
- Card
- collated
- laminate (a4)
- laminat (a3)
- Coloured paper
- plain paper
- bw print
- colour print
- Two hole punched

Change/add print credits



Validation to check that all boxes have been filled will be added.

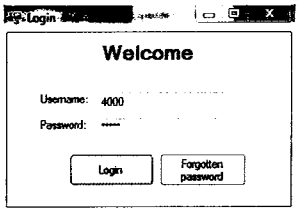
Section 3

Implementation

Screen shots

Login screen

The login screen shown below accomplishes objective 2 in the anaylsis section. The user details are stored in an XML file and using the 'user' class the file is opened and checked to see if it is matched by what the user has entered.



The code shown below shows what happens when the login button is pressed:

```
PrivateSubloginbutton_Click(ByVal sender AsSystem.Object, ByVal e AsSystem.EventArgs)
Handlesloginbutton.Click
DimloggingdetailsAsNewgetenteredlogindetails
DimvalidloginAsNewfinduser
DimusernamepasswordAsString
DimcheckuserAsBoolean
loggingdetails.username = username.Text
loggingdetails.password = password.Text
usernamepassword = loggingdetails.username&loggingdetails.password

checkuser = validlogin.userpasswordfound(usernamepassword)

Ifcheckuser = TrueThen
Ifloggingdetails.username = 8000Then
MsgBox("Welcome to the system")
Reprographics_usermenu.Show()
Me.Visible = False
Else
MsgBox("Welcome to the system")
User_Menu.Show()
Me.Visible = False
EndIf
ElseIfcheckuser = FalseThen
MsgBox("Invalid username/password")
EndIf

EndSub
```

Comment [C3]: UserID and Password saved to this variable

The code shown below shows part of the 'userprofile class that is used to open the userdetails XML file:

```
Functionuserid(ByVal number AsInteger)
DimuserdetailsAsNewSystem.Xml.XmlDocument
userdetails.Load("userdetails.xml")
DimreaduserdetailsAsSystem.Xml.XmlNodeList = userdetails.SelectNodes("users/user")
userid = readuserdetails.Item(number).SelectSingleNode("userid").InnerText()

EndFunction
```

Comment [C4]: Defines variable as an XML file.

Comment [C5]: This is the file name that will be used to extracte and save the information

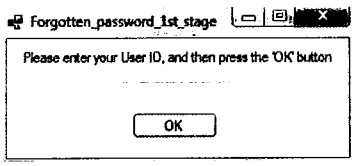
Comment [C6]: Name of child node

Part of the contents of the XML file that is used to store the user data is shown below:

```
<user>
<userid>2202</userid>
<department>D of E</department>
<password>2202</password>
<confirmpassword>2202</confirmpassword>
<securityquestion>What is your name?</securityquestion>
<securityanswer>2202</securityanswer>
</user>
```

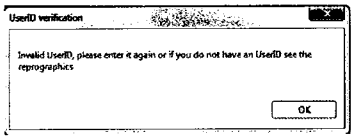
Forgotten password 1st stage

If the user has forgotten their password they can click on the 'fogotten password' screen which is on the login screen, when they click on the 'forgotten password' button they will see the screen below, this is the first stage of the security. When the user then presses the 'ok' button,two different messages can appear and these are shown in screenshots (A) and (B) below:



(A) This message will appear is if the userID is a valid one.

(B) This is the message that will appear if the UserID entered is invalid.




```
PublicClassForgotten_password_1st_stage

PrivateSub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Dim userfound AsNewfinduser
Dim checkuser AsBoolean
Dim entereduserid AsNewgetenteredlogindetails
Dim secondstageofforgottenpassword AsNewforgotten_password_final_stage

        entereduserid.username = UserID.Text
        checkuser = userfound.userfound(entereduserid.username)
If checkuser = TrueThen
MsgBox("User ID verified, please enter your security answer", MsgBoxStyle.OkOnly,
"UserID verification")
forgotten_password_final_stage.Show()
Me.Close()

ElseIf checkuser = FalseThen
```

Comment [C7]: Creates message box

Comment [C8]: What the message box should say

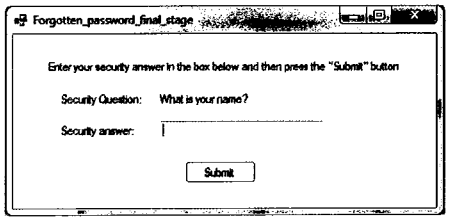
Comment [C9]: Type of message box, in this case the end user was happy with just an OK button

Comment [C10]: Name of message box

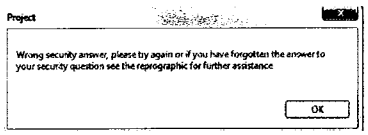
```
            MsgBox("Invalid UserID, please enter it again or if you do not have an
UserID see the reprographics", MsgBoxStyle.OkOnly, "UserID verification")
EndIf

EndSub
EndClass
```

Forgotten password final stage

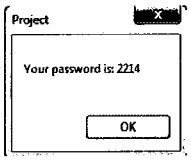
When the user then presses the 'ok' button, two different messages can appear and these are shown in screenshots (C) and (D) below:



(C) If the answer to the security question is wrong then the following message will appear:

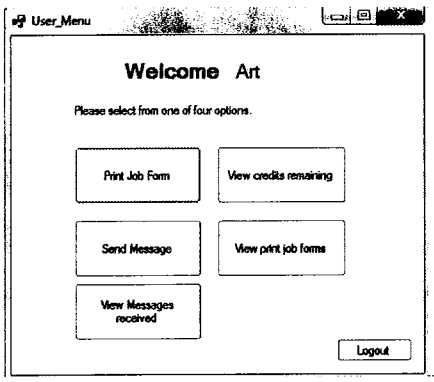


(D) If the answer to the security answer is correct then the following message will appear,:



The class 'userprofile' is used to open the XML file and extract the information required from the file.

User main menu



Comment [C11]: The subject title (in case it is Art) written after the 'Welcome' depends on the User that has logged on so depending on the UserID entered in the login screen, the subject title changes according to what department that UserID corresponds to.

Comment (C17) shows where the code for deciding which subject title should be written. For example for UserID '2214' will write 'Economics' as UserID '2214' corresponds to the Economics department.

The 'Logout' button shown on the screen shot of the user main menu above is cross referenced to objective 10 in the Analysis section. The user main menu agrees with objective 11.

```
PublicClassUser_Menu

PrivateSub sendmessage_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles sendmessage.Click
Send_Message.Show()
Me.Close()
EndSub
PrivateSub printjobform_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles printjobform.Click
Print_job_form.Show()
Me.Close()
EndSub

PrivateSub creditremaining_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles viewcreditsremaining.Click
Credit_remaining.Show()
Me.Close()
EndSub

PrivateSub viewprintforms_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles viewprintjobforms.Click
View_print_forms.Show()
Me.Close()
EndSub
PrivateSub logout_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles logout.Click
        Login.username.Clear()
        Login.password.Clear()
Login.Show()
Me.Close()
EndSub

PrivateSubUser_Menu_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
HandlesMyBase.Load
Dim finddepartmentname AsNewfinduser
Dim getinformation AsNewcheckuserdata
Dim useridAsInteger
userid = finddepartmentname.finduserprofile(Login.username.Text)
Label.Text = getinformation.department(userid)
EndSub

PrivateSub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
usercheckmessages.Show()
Me.Close()
EndSub
EndClass
```

Comment [C12]: This closes the current window which in this case is the User main menu.

Comment [C13]: This is the screen that it will open

Comment [C14]: This means this private sub is triggered when some clickes on the button labelled 'View credits remaining'.

Comment [C15]: This Private sub is triggered and implemented when the User main meny loads.

Comment [C16]: This opens the 'find user' class and searches the XML file 'user details' to find the user profile child node number of the user that has just logged on, the child number will be used to select theright information in the file.

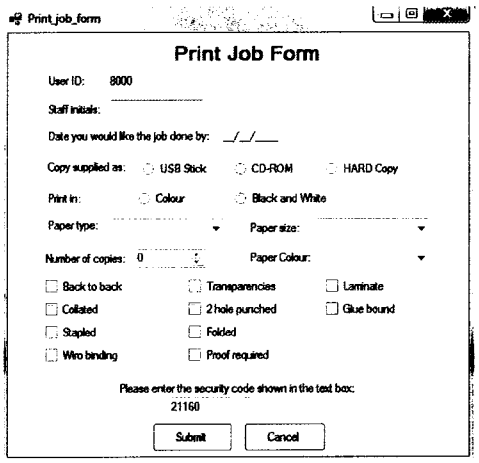
The child node number will be saved to the variable usernumber.

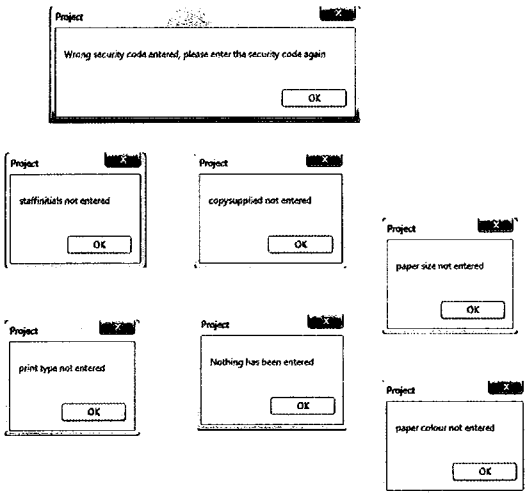
The information wanted is to see which department the user that just logged on is so I can be used to put the correct subject tiltle can be shown.

Comment [C17]: The usernumber will be used to extract the required information from the XML file (user details) and save it to the subject label so that it is shown to the user once the user main menu is shown.

Print job form

Below shows a screenshot of the print job form and screenshots next to it show the possible error messages that the user can be faced with. This is cross-referenced to objectives 5, 6, 7, 14, 17 and 22 stated in the analysis section. Objective 6 is shown in the code below the screenshot. There is a security code at the bottom which the user has to enter in before the print job form can be submitted. The security code is generated randomly (objective 21).






```
PrivateSubsubmitted_Click(ByVal sender AsSystem.Object, ByVal e AsSystem.EventArgs)
Handlessubmitted.Click
Dim totalprice AsInteger
Dim gettotalprice AsNewcalculate_total_price_of_print_job
Dim accountdetails AsNewaccounts
Dim confirm AsNewprintcredits
Dim canjobproceed AsBoolean = False
Dim laminatea3 AsBoolean
Dim laminatea4 AsBoolean
Dim check AsNewprintjobform
Dim jobnumber AsInteger = check.Length
Dim checkuser AsNewfinduser
Dim uservalid AsBoolean = False
Dim sendprintjobform AsBoolean = True
Dim department AsString
Dim usernumber AsInteger
Dim user AsNewfinduser

uservalid = checkuser.userfound(userid.Text)
usernumber = user.finduserprofile(userid.Text)
department = user.finddepartment(usernumber)

Ifsecuritycodeentered.Text = securitycode.TextAnduservalid = TrueThen
jobnumber = jobnumber + 1
Ifuserid.Text<>""Then
newform.userid = userid.Text
Else
sendprintjobform = False
```

Comment [C18]: Checks to see if userID entered is a valid.

Comment [C19]: Validation to see f the security code the user has entered matches the security code number shown. Will output an error message if it does not match. (see comment (029)

```
MsgBox("userid not entered")
EndIf
If TextBox1.Text <>""Then
newform.staffinitials = TextBox1.Text
Else
sendprintjobform = False
MsgBox("staffinitials not entered")
EndIf
If MaskedTextBox1.Text <>""Then
newform.requestdate = MaskedTextBox1.Text
Else
sendprintjobform = False
MsgBox("Request date not entered")
EndIf
Ifcdrom.Checked = TrueThen
newform.copysuppliedas = cdrom.Text
ElseIfusbstick.Checked = TrueThen
newform.copysuppliedas = usbstick.Text
ElseIfhardcopy.Checked = TrueThen
newform.copysuppliedas = hardcopy.Text
ElseIfcdrom.Checked = FalseAndusbstick.Checked = FalseAndhardcopy.Checked = FalseThen
sendprintjobform = False
MsgBox("copysupplied not entered")
EndIf
Ifcolour.Checked = TrueThen
newform.printtype = colour.Text
ElseIfbandw.Checked = TrueThen
newform.printtype = bandw.Text
ElseIfcolour.Checked = FalseAndbandw.Checked = FalseThen
sendprintjobform = False
MsgBox("print type not entered")
EndIf
If ComboBox2.Text <>""Then
newform.papertype = ComboBox2.Text
Else
sendprintjobform = False
EndIf
If ComboBox3.Text <>""Then
newform.papersize = ComboBox3.Text
Else
sendprintjobform = False
MsgBox("paper size not entered")
EndIf

If NumericUpDown1.Value <= 700Then
newform.numberofcopies = NumericUpDown1.Text
canjobproceed = confirm.updateprintcredits(newform.userid, newform.numberofcopies)
Else
MsgBox("You have gone over the maximum limit which is 700 copies, please input number
700 and below")
EndIf
If ComboBox1.Text <>""Then
newform.papercolour = ComboBox1.Text
Else
sendprintjobform = False
MsgBox("paper colour not entered")
EndIf
newform.backtoback = CheckBox1.Checked
newform.collated = CheckBox2.Checked
newform.stapled = CheckBox3.Checked
newform.wirobinding = CheckBox4.Checked
newform.laminate = CheckBox5.Checked
```

Comment [C20]: Validation: checks to see if everything has been filled in and if hot then error messages will be outputted to the user

Comment [C21]: Checks to see if number of copies entered is 700 and less and if not an error message is outputted to the user.

Comment [C22]: Checks to see if the user has enough print credits for the job to proceed. This uses the a function 'update print credits'. If the user has enough credits then the Boolean variable 'canjobproceed' is true and if the user doesn't have enough credits then it will say 'false'.

Comment [C23]: Error message and what the error message says

Comment [C24]: Validation checks to see if paper colour has been entered and if not an error message is outputted.

```
Ifnewform.laminate = TrueAndnewform.papersize = "a3"Then
                laminatea3 = True
ElseIfnewform.laminate = TrueAndnewform.papersize = "a4"Then
                laminatea4 = True
EndIf
newform.gluebound = CheckBox6.Checked
newform.transparencies = CheckBox12.Checked
newform.twoholepunched = CheckBox11.Checked
newform.folded = CheckBox10.Checked
newform.proofrequired = CheckBox9.Checked
Ifcanjobproceed = TrueAndsendprintjobform = TrueThen
totalprice = gettotalprice.calculatetotalprice(newform.papertype, newform.papersize,
newform.printtype, newform.papercolour, newform.backtoback, newform.collated,
newform.stapled, newform.wirobinding, laminatea4, laminatea3, newform.gluebound,
newform.transparencies, newform.twoholepunched, newform.folded, newform.numberofcopies)
job.Add(jobnumber, newform.userid, newform.staffinitials, newform.requestdate,
newform.copysuppliedas, newform.printtype, newform.papertype, newform.papersize,
newform.numberofcopies, newform.papercolour, newform.backtoback, newform.collated,
newform.stapled, newform.wirobinding, newform.laminate, newform.gluebound,
newform.transparencies, newform.twoholepunched, newform.folded, newform.proofrequired)
accountdetails.Add(jobnumber, newform.userid, newform.staffinitials, department,
newform.requestdate, newform.numberofcopies, totalprice)
If Login.username.Text = "8000"Then
                    MsgBox("Print job form submitted successfully")
Reprographics_usermenu.Show()
Me.Close()

Else
                    MsgBox("Print job form submitted successfully")
User_Menu.Show()
Me.Close()
EndIf
ElseIf canjobproceed = FalseThen
                MsgBox("Print job unsuccessful because you have insufficient print
credits")
ElseIf totalprice = 0Then
                MsgBox("Nothing has been entered")
EndIf
ElseIf uservalid = FalseThen
            MsgBox("UserID invalid, please enter a valid UserID")
ElseIf securitycodeentered.Text <> securitycode.Text Then
            MsgBox("Wrong security code entered, please enter the security code again")

EndIf

EndSub
```

Comment [C25]: Saves true or false to variable newform.gluebound which uses a 'property get' function to collect function. For all the check boxes the data type is a Boolean, 'true' if the check box is checked and 'false' is the check box is unchecked.

Comment [C26]: Calculates total cost of job, passes in the variables that contains the information from the print job form. And then saves this to the variable 'totaltalprice'.

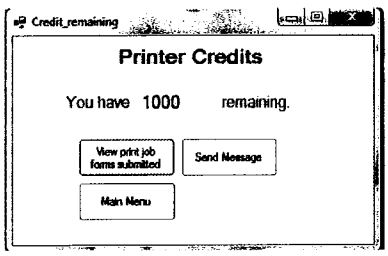
Comment [C27]: Passes variables that contains all the information from the print job form to the 'save print job form' to the XML file 'printjobform'.

Comment [C28]: Passes variables that contains all the information from the print job form to the 'save print job form' to the XML file 'accoutndetails'.

Comment [029]: Error message to say that security message has not been entered properly.

Credit remaining

The screenshot below is based on objective 28 with the code below it showing how it has been done

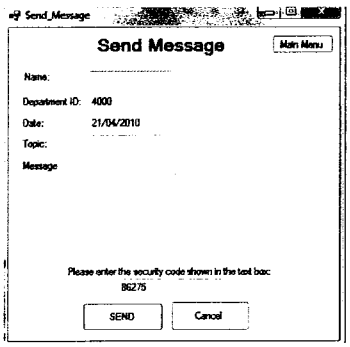


```
PrivateSub Credit_remaining_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) HandlesMyBase.Load
Dim printcredits AsNewprintcredits
Dim findusernumber AsNewfinduser
Dim usernumber AsInteger
        usernumber = findusernumber.finduserprofile(Login.username.Text)
        creditsremaining.Text = printcredits.printcredits(usernumber)
EndSub
```

Comment [C30]: Class used 'print credits'

Send Message

The send message screen shows the User ID and date automatically, this only done for the user 'send message' screen. For the reprographics the userID will have to be filled in. This is shown on the 'reprographics send message' screenshot. There is a security code at the bottom which the users has to enter in before the message can be submitted. The security code is generated randomly (objective 22). The code for how this is done is shown below:




```
PrivateSub Send_Message_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) HandlesMyBase.Load
Dim randomnumber AsNewRandom
Dim x AsInteger
        x = randomnumber.Next(1, 1000000)
        securitycode.AppendText(x)
        userid.Text = Login.username.Text
        todaysdate.Text = Now.Date
EndSub
```

Comment [C31]: This generates the random security number when the 'send message' screen opens.

The code below shows the process that happens when the 'Send' button is pressed:

```
PrivateSub send_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles send.Click
If Login.username.Text = "8000"Then
Dim check AsNewreprographics_sent_messages
Dim getdetails AsNewgetmessagedetails
Dim savedetails AsNewreprographics_sent_messages
Dim messagenumber AsInteger = check.Length
Dim sendmessage AsBoolean = True
If securitycodeentered.Text = securitycode.Text Then
                messagenumber = messagenumber + 1

If message.Text <>""Then
                    getdetails.getmessage = message.Text
```

```
                    sendmessage = False
Else
                    MsgBox("You have not entered your message")
EndIf

If sendmessage = TrueThen
savedetails.Add(messagenumber, getdetails.getuserid, getdetails.getname,
getdetails.getdatesubmitted, getdetails.gettopic, getdetails.getmessage)
                    MsgBox("Message sent to reprographics successfully")
User_Menu.Show()
Me.Close()
EndIf
Else
                MsgBox("Wrong or no security code entered, please enter the security
code again")
EndIf
Else
Dim messageclass AsNewuser_sent_message
Dim getmessagedetails AsNewgetmessagedetails
Dim findmessagenumber AsInteger = messageclass.length
Dim sendthismessage AsBoolean = False
If securitycodeentered.Text = securitycode.Text Then
                findmessagenumber = findmessagenumber + 1
If nameentered.Text <>""Then
                    getmessagedetails.getname = nameentered.Text
                    sendthismessage = True
Else
                    MsgBox("You have not entered your name")
                    sendthismessage = False
EndIf
If topicentered.Text <>""Then
                    getmessagedetails.gettopic = topicentered.Text
                    sendthismessage = True
Else
                    MsgBox("You have not entered the topic")
                    sendthismessage = False
EndIf
If message.Text <>""Then
                    getmessagedetails.getmessage = message.Text
                    sendthismessage = True
Else
                    MsgBox("You have not entered the message")
                    sendthismessage = False
EndIf
                getmessagedetails.getdatesubmitted = todaysdate.Text
                getmessagedetails.getuserid = userid.Text
If sendthismessage = TrueThen
                    savemessagedetails.Add(findmessagenumber,
getmessagedetails.getuserid, getmessagedetails.getname,
getmessagedetails.getdatesubmitted, getmessagedetails.gettopic,
getmessagedetails.getmessage)
                    MsgBox("Message sent to reprographics successfully")
User_Menu.Show()
Me.Close()
EndIf
Else
                MsgBox("Wrong or no security code entered, please enter the security
code again")
EndIf


EndIf
```

Comment [C32]: Passses variables that contain the detaiks of the message (like topic, userID, date and message) to the function 'save message details' which saves the information to the 'reprographics sent messages' XML file.

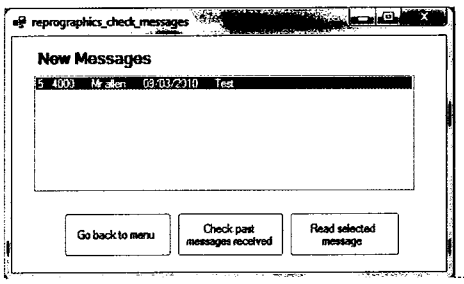
Comment [C33]: This is an example a validation check. This validation checks to see if the name of the person sending the message has been entered.

Comment [C34]: Passses variables that contain the detaiks of the message (like topic, userID, date and message) to the function 'save message details' which saves the information to the 'usermessages' XML file.

```
EndSub
```

Screenshots 'Reprographics check new messages', 'Reprographics check past messages', 'view new messages (user)' and 'View past messages (user)' cross reference to objective 23 and the coding that enables the process to happen is shown below each.

Reprographics check new messages

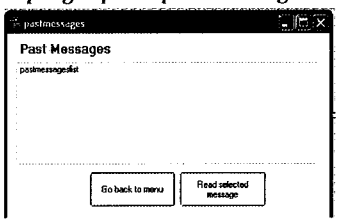


Comment [C35]: The screen shows that there is one message in currently in the stack.

```
PublicClassreprographics_check_messages

PrivateSub gobacktomenu_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles gobacktomenu.Click
Reprographics_usermenu.Show()
Me.Close()
EndSub
PrivateSub reprographics_check_messages_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) HandlesMyBase.Load
Dim update AsNewaddtoqueue
update.addnewmessagetoqueue()
EndSub

PrivateSub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
Dim messageread AsNewuser_sent_message
Dim message AsString
        message = newmessagestack.SelectedItem
        messageread.setmessagetoread(message)
view_selected_new_message.Show()
Me.Close()
EndSub

PrivateSub pastmessages_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles checkpastmessages.Click
Reprographics_past_messages.Show()
Me.Close()
EndSub
EndClass
```

Comment [C36]: Uses 'queue' class to add any messages that has not been read yet to the class.

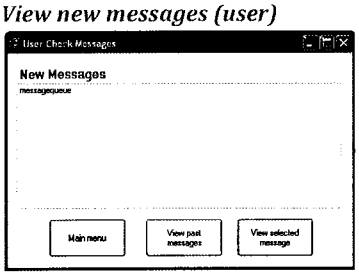
Reprographics past messages




```
PublicClassReprographics_past_messages

PrivateSub pastmessages_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) HandlesMyBase.Load
Dim update AsNewaddtoqueue
update.addpastmessagetoqueue()
EndSub

PrivateSub mainmenu_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles mainmenu.Click
Reprographics_usermenu.Show()
Me.Close()
EndSub

PrivateSub readselectedmessage_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles readselectedmessage.Click
Dim messageread AsNewuser_sent_message
view_selected_past_message.Show()
      messageread.setmessagetoread(pastmessagesstack.SelectedValue)
Me.Close()
EndSub
EndClass
```

Comment [C37]: Adds all messages that have been addressed to the reprographics to the message stack .

View new messages (user)

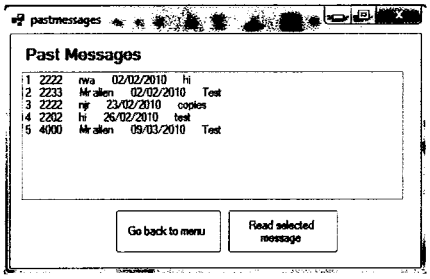



```
PublicClassusercheckmessages

PrivateSub mainmenu_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles mainmenu.Click
User_Menu.Show()
Me.Close()
EndSub

PrivateSub usercheckmessages_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) HandlesMyBase.Load
Dim update AsNewaddtoqueue
update.addusernewmessagetoqueue(Login.username.Text)
EndSub
```

Comment [C38]: Adds any new messages received to the user by the reprographics that has not been read and adds it to the message stack.

```
PrivateSub viewselectedmessage_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles viewselectedmessage.Click
view_selected_message_user_.Show()
EndSub

PrivateSub viewpastmessages_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles viewpastmessages.Click
usercheckpastmessages.Show()
Me.Close()
EndSub
EndClass
```

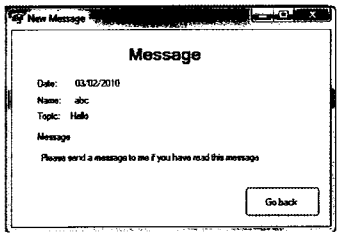
User check past messages




```
PublicClassusercheckpastmessages

PrivateSub mainmenu_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles mainmenu.Click
User_Menu.Show()
Me.Close()
EndSub

PrivateSub usercheckpastmessages_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) HandlesMyBase.Load
Dim update AsNewaddtoqueue
update.adduserpastmessagetostack()
EndSub

PrivateSub viewselectedmessage_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles viewselectedmessage.Click
      view_selected_new_message_user_.Show()
Me.Close()

EndSub
EndClass
```

Comment [C39]: Adds allmessages received that the user has received and adds it to the message stack.

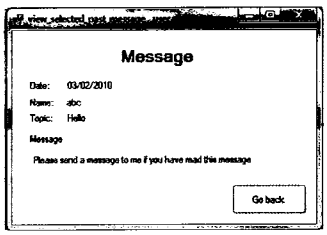
View selected new message (user)



```
PublicClassview_selected_new_message_user_

PrivateSub view_selected_message_user__Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) HandlesMyBase.Load
Dim messagedetails AsNewreprographics_sent_messages
Dim number AsInteger
Dim find AsNewfindjob
Dim selectedjob AsString
        selectedjob = usercheckpastmessages.messagestack.SelectedItem
        number = find.findrightmessageinuserlogin(selectedjob)
        nameentered.Text = messagedetails.name(number)
        datesubmitted.Text = messagedetails.datesubmitted(number)
        topic.Text = messagedetails.topic(number)
        message.Text = messagedetails.submittedmessage(number)
EndSub

PrivateSub cancel_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles cancel.Click
usercheckpastmessages.Show()
Me.Close()
EndSub
EndClass
```

View selected past message (user)

The screenshot shown below agrees with objective 27.

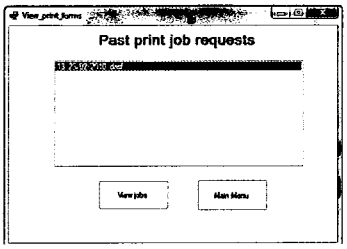



```
PublicClassview_selected_message_user_

PrivateSub view_selected_past_message__user__Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) HandlesMyBase.Load

Dim messagedetails AsNewreprographics_sent_messages
Dim number AsInteger
Dim find AsNewfindjob
Dim selectedjob AsString
        selectedjob = usercheckpastmessages.messagepastmessagestack.SelectedItem
        number = find.findrightmessageinuserlogin(selectedjob)
        nameentered.Text = messagedetails.name(number)
        datesubmitted.Text = messagedetails.datesubmitted(number)
        topic.Text = messagedetails.topic(number)
        message.Text = messagedetails.submittedmessage(number)
EndSub

PrivateSub cancel_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles cancel.Click
usercheckpastmessages.Show()
Me.Close()
EndSub
EndClass
```

Comment [C40]: Function extracts information from XML file named 'usermessages' and outputs the information to the user.

View past jobs

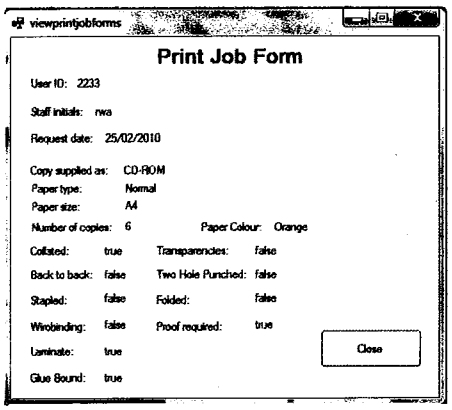
The screenshot below allows the user to view past print job fotms they have sent (objective 30) and the 'view selected past print job form screen' allows the user to view the selected job, (Objective 34).




```
PublicClassView_print_forms

PrivateSub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
User_Menu.Show()
Me.Close()
EndSub

PrivateSub View_print_forms_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) HandlesMyBase.Load
Dim update AsNewaddtoqueue
        update.addtolistpastjobs()
EndSub

PrivateSub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
viewpastprintjobforms.Show()
Me.Close()
EndSub

EndClass
```

View selected past print job form

This allows the user to view the selected past job in more detail.



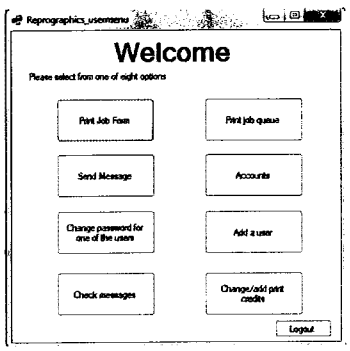

```
PublicClassviewprintjobforms
```

```
PrivateSub viewprintjobforms_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) HandlesMyBase.Load
Dim viewpastjobs AsNewviewingpastjobs
Dim number AsInteger
Dim jobnumber AsNewPrint_job_queue
Dim find AsNewfindjob
Dim selectedjob AsString
        selectedjob = Print_job_queue.Queue.SelectedItem
        number = find.findrightjob(selectedjob)
        number = number
        UserID.Text = viewpastjobs.userid(number)
        Staffinitials.Text = viewpastjobs.staffinitials(number)
        Requestdate.Text = viewpastjobs.requestdate(number)
        copysuppliedas.Text = viewpastjobs.copysuppliedas(number)
        papertype.Text = viewpastjobs.papertype(number)
        papersize.Text = viewpastjobs.papersize(number)
        numberofcopies.Text = viewpastjobs.numberofcopies(number)
        papercolour.Text = viewpastjobs.papercolour(number)
        backtoback.Text = viewpastjobs.backtoback(number)
        collated.Text = viewpastjobs.collated(number)
        stapled.Text = viewpastjobs.stapled(number)
        wirobinding.Text = viewpastjobs.wirobinding(number)
        laminate.Text = viewpastjobs.laminate(number)
        gluebound.Text = viewpastjobs.gluebound(number)
        transparencies.Text = viewpastjobs.transparencies(number)
        twoholepunched.Text = viewpastjobs.twoholepunched(number)
        folded.Text = viewpastjobs.folded(number)
        proofrequired.Text = viewpastjobs.proofrequired(number)
EndSub

PrivateSub closebutton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles closewindow.Click
Print_job_queue.Show()
Me.Close()
EndSub

EndClass
```

Comment [C41]: Function extracts information from XML file named 'printj job form' and outputs the information to the user.

Reprographics main menu

The screen shot below is shown to the reprographics when they log on to the system, (objective 14)




```
PublicClassReprographics_usermenu

PrivateSub printjobform_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles printjobform.Click
```

```
Print_job_form.Show()
Me.Close()
EndSub

PrivateSub printjobqueue_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles printjobqueue.Click
Print_job_queue.Show()
Me.Close()
EndSub

PrivateSub accounts_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles accounts.Click
Reprographics_Accounts.Show()
Me.Close()
EndSub

PrivateSub addauser_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles addauser.Click
Add_a_user.Show()
Me.Close()
EndSub

PrivateSub changepasswordforoneoftheusers_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles changepasswordforoneoftheusers.Click
change_username_password_of_user.Show()
Me.Close()
EndSub

PrivateSub checkmessages_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles checkmessages.Click
reprographics_check_messages.Show()
Me.Close()
EndSub

PrivateSub changeaddprintcredits_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles changeaddprintcredits.Click
change_add_print_credits.Show()
Me.Close()
EndSub

PrivateSub sendamessage_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles sendamessage.Click
send_message__reprographics_.Show()
Me.Close()
EndSub

PrivateSub logout_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles logout.Click
        Login.username.Clear()
        Login.password.Clear()
Login.Show()
Me.Close()
EndSub

PrivateSub Reprographics_usermenu_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) HandlesMyBase.Load

EndSub
EndClass
```

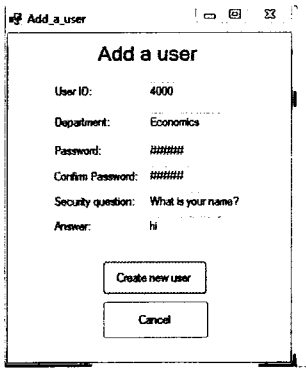
Comment [i42]: Opens the page stated

Comment [i43]: Closes current page

Comment [C44]: When someone logs out, the username and password is automatically cleared so any information stored about the user that has logged on is cleared.

Add a user

This allows the reprographics to add a new user to the system and at the same time a print credit profile is created. (objective 24)



Comment [C45]: Password and confirm password has been validated so that no one can see what the password is if anything is watching when the password is being typed.

```
PublicClassAdd_a_user

PrivateSub cancel_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles cancel.Click
Reprographics_usermenu.Show()
Me.Close()
EndSub

PrivateSub createnewuser_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles createnewuser.Click
Dim getdetails AsNewgettinguserdetails
Dim saveuserdetails AsNewuserprofile
Dim passwordcheck AsBoolean
Dim printcredits AsNewprintcredits
        getdetails.userid = MaskedTextBox1.Text
        getdetails.department = MaskedTextBox2.Text

        getdetails.password = MaskedTextBox3.Text
        getdetails.confirmpassword = MaskedTextBox4.Text
        getdetails.securityquestion = TextBox1.Text
        getdetails.securityanswer = TextBox2.Text

If getdetails.password = getdetails.confirmpassword Then
            passwordcheck = True
            printcredits.saveprintcredits(getdetails.userid, 1000)
            saveuserdetails.saveuser(getdetails.userid, getdetails.department,
getdetails.password, getdetails.confirmpassword, getdetails.securityquestion,
getdetails.securityanswer)
            MsgBox("User Successfully added", MsgBoxStyle.OkOnly, "Add a user")
Else
            MsgBox("Password and confirm password are not the same, please enter them
again")
            passwordcheck = False
EndIf
EndSub

EndClass
```

Comment [C46]: Information that has been inputted in the 'add a user' form is save to variables. This uses the 'property get' procedure to collects and save the information to the variable.

Comment [C47]: Steps that are undergone when the 'create new user' button has been pressed.

The code below adds a print job profile for the user automatically once added:

```
printcredits.Add(getdetails.userid, 1000)
```

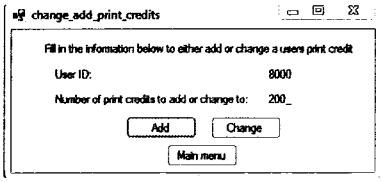
Comment [C48]: Creates a print credit profikle for user. (See comment (C40))

```
<printcredit>
<userid>1111</userid>
<printcredits>9999</printcredits>
</printcredit>
```

Comment [C49]: Print credit profile

Change/add print credits

The following screenshot allows the reprographics to add or change a user print credits, (objective 29).




```
PublicClasschange_add_print_credits

PrivateSub mainmenu_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles mainmenu.Click
Reprographics_usermenu.Show()
Me.Close()
EndSub

PrivateSub add_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles add.Click
Dim changeadd AsNewprintcredits
Dim details AsNewgetdetailsforprintcreditchange
Dim findusernumber AsNewfinduser
Dim usernumber AsInteger
Dim currentnumberofcredits AsInteger
Dim newprintcredits AsInteger

        details.getuserid = userid.Text
        details.getnumberofprintcredits = numberofprintcredits.Text

        usernumber = findusernumber.finduserprofile(details.getuserid)
        currentnumberofcredits = changeadd.lookupprintcredits(usernumber)
        newprintcredits = currentnumberofcredits + details.getnumberofprintcredits
        changeadd.changeprintcredits(usernumber, newprintcredits)
EndSub
```

Comment [C50]: Adds the print credits specified by the reprograohics to the initial value

```
PrivateSub change_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles change.Click
Dim changeadd AsNewprintcredits
Dim details AsNewgetdetailsforprintcreditchange
Dim findusernumber AsNewfinduser
Dim usernumber AsInteger

        details.getuserid = userid.Text
        details.getnumberofprintcredits = numberofprintcredits.Text

        usernumber = findusernumber.finduserprofile(details.getuserid)
        changeadd.changeprintcredits(usernumber, details.getnumberofprintcredits)
EndSub
EndClass
```

Comment [C51]: Changes the value of the print credits to the amount specified by the user.

Change password

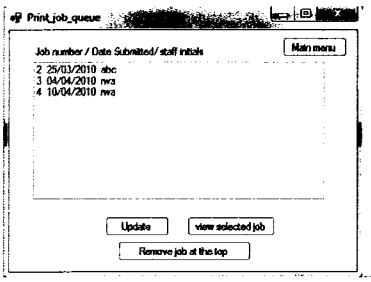
This allows the reprographics to change the password for the user.


```
PrivateSub changepassword_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles changepassword.Click

Dim userid AsString
Dim finduserdetails AsNewfinduser
Dim entereduserid AsNewgetenteredlogindetails
Dim currentpasswordentered AsString
Dim newpassword AsString
Dim newconfirmedpassword AsString
Dim usernumber AsInteger
Dim userprofile AsNewuserprofile

        userid = Confirm_user_ID.UserID.Text & Confirm_user_ID.currentpassword.Text
        usernumber = finduserdetails.findusersequrityquestion(userid)
        currentpasswordentered = MaskedTextBox2.Text
        newpassword = MaskedTextBox3.Text
        newconfirmedpassword = MaskedTextBox4.Text
userprofile.changepassword(usernumber, newpassword)
EndSub
```

Comment [C52]: Function changes user intial password to what new password.

Print job queue

This allows the reprographics to see all the jobs that need to be done which have been submitted by the users and the system has validated the job to see if there are any problems and if not then it is added to the print job queue (objective 6), it is sorted by date the job needs to be done making it a priortised queue.



Comment [i53]: By pressing the 'update' button, it clears the queue, initialises it again, adds contents to the queue and then outputs to contents.

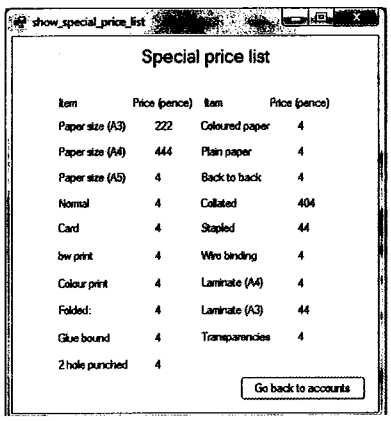
```
PublicClassPrint_job_queue

PrivateSub update_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles updatequeue.Click
Dim update AsNewaddtoqueue
        update.addformtoqueue()
EndSub
```

```
PrivateSub viewselectedjob_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles viewselectedjob.Click
viewprintjobforms.Show()
EndSub

PrivateSub Print_job_queue_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) HandlesMyBase.Load
Dim update AsNewaddtoqueue
        update.addformtoqueue()
EndSub

PrivateSub mainmenu_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles mainmenu.Click
Reprographics_usermenu.Show()
Me.Close()
EndSub

PrivateSub removejob_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles removejob.Click
Dim topofstack AsString
Dim changeprintjobformstatus AsNewprintjobform
Dim topstack AsNewaddtoqueue
        topofstack = topstack.addformtoqueue()
        changeprintjobformstatus.jobcompleted(topofstack)
        Queue.Items.Remove(topofstack)
EndSub
EndClass
```

Comment [C54]: This is what happens when the form 'print job queue' laods.

Special price list

This window shows all the prices (in pence) of the special prices. (objective 37)



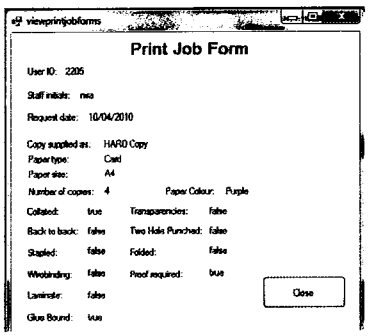

```
PrivateSub show_special_price_list_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) HandlesMyBase.Load
Dim lookupprice AsNewspecial_price_list
        papersizea3.Text = lookupprice.papersizea3
        papersizea4.Text = lookupprice.papersizea4
        papersizea5.Text = lookupprice.papersizea5
        normal.Text = lookupprice.normal
        card.Text = lookupprice.card
        bwprint.Text = lookupprice.bwprint
        colourprint.Text = lookupprice.colourprint
        folded.Text = lookupprice.folded
        gluebound.Text = lookupprice.gluebound
```

```
        twoholepunched.Text = lookupprice.twoholepunched
        colouredpaper.Text = lookupprice.papercolour
        plainpaper.Text = lookupprice.plainpaper
        backtoback.Text = lookupprice.backtoback
        collated.Text = lookupprice.collated
        stapled.Text = lookupprice.stapled
        wirobinding.Text = lookupprice.wirobinding
        laminatea4.Text = lookupprice.laminatea4
        laminatea3.Text = lookupprice.laminatea3
        transparencies.Text = lookupprice.transparencies
EndSub
```

Comment [C55]: Uses class 'look up special prices' to extract information about the price and output to the reprograhics.

View selected job

This is the job shown in more detail when the reprographics selects a job from the print job queue. (objective 31)



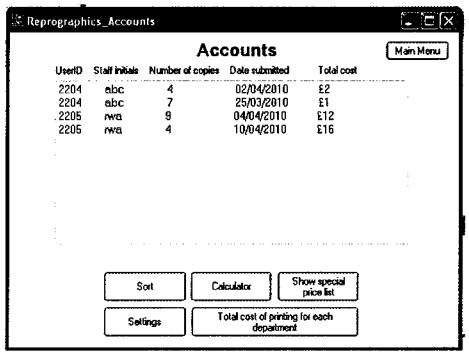

```
PublicClassviewprintjobforms
PrivateSub viewprintjobforms_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) HandlesMyBase.Load
Dim viewpastjobs AsNewprintjobform
Dim number AsInteger
Dim jobnumber AsNewPrint_job_queue
Dim find AsNewfindjob
Dim selectedjob AsString
        selectedjob = Print_job_queue.Queue.SelectedItem
        number = find.findrightjob(selectedjob)
        number = number
        UserID.Text = viewpastjobs.userid(number)
        Staffinitials.Text = viewpastjobs.staffinitials(number)
        Requestdate.Text = viewpastjobs.requestdate(number)
        copysuppliedas.Text = viewpastjobs.copysuppliedas(number)
        papertype.Text = viewpastjobs.papertype(number)
        papersize.Text = viewpastjobs.papersize(number)
        numberofcopies.Text = viewpastjobs.numberofcopies(number)
        papercolour.Text = viewpastjobs.papercolour(number)
        backtoback.Text = viewpastjobs.backtoback(number)
        collated.Text = viewpastjobs.collated(number)
        stapled.Text = viewpastjobs.stapled(number)
        wirobinding.Text = viewpastjobs.wirobinding(number)
        laminate.Text = viewpastjobs.laminate(number)
        gluebound.Text = viewpastjobs.gluebound(number)
        transparencies.Text = viewpastjobs.transparencies(number)
        twoholepunched.Text = viewpastjobs.twoholepunched(number)
        folded.Text = viewpastjobs.folded(number)
        proofrequired.Text = viewpastjobs.proofrequired(number)
EndSub
```

Comment [i56]: Using 'print job form' class to extract information from the XML file and save it to the variables.

```
PrivateSub closebutton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles closewindow.Click
Print_job_queue.Show()
Me.Close()
EndSub

EndClass
```

Accounts

The accounts shows the cost details of each print job submitted by the user. (Objective 19).

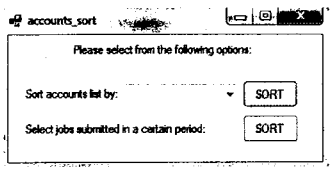


The account's class is used to extract the accounts information required to add to the accounts stack, part of the code for the accounts is shown below:

```
Function userid(ByVal number AsInteger)
Dim accounts AsNew System.Xml.XmlDocument
        accounts.Load("accountdetails.xml")
Dim accountdetails As System.Xml.XmlNodeList =
accounts.SelectNodes("accounts/useraccountdetails")
        userid = accountdetails.Item(number).SelectSingleNode("userid").InnerText()
EndFunction
```

Accounts sort

This gives the option for the reprographics to either sort the accounts by UserID, date job submitted and total cost or to select jobs submitted in a certain period. (Objectives 32, 33)

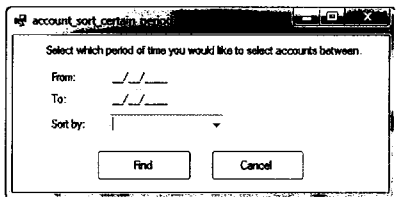



```
PublicClassaccounts_sort

PrivateSub sortlist_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles sortlist.Click
accounts_sort_list_by.Show()
Me.Visible = False
EndSub
```

```
PrivateSub sortcertainperiod_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles sortcertainperiod.Click
account_sort_certain_period.Show()
Me.Close()
EndSub

EndClass
```

Accounts sort certain period

This window is used to select the dates you want to view past jobs and also you can then sort this data by UserID, date submitted or total cost.

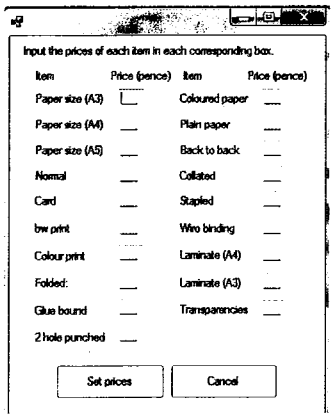



```
PublicClassshow_accounts_certain_period

PrivateSub show_accounts_certain_period_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) HandlesMyBase.Load

Dim length AsInteger
Dim number AsInteger = 0
Dim accounts AsNewaccounts
Dim type AsString
        type = accounts_sort.sortoptions.Text
        length = accounts.Length
Dim completed AsBoolean = True
Dim temporystoredvalue AsInteger
Dim array(length) AsInteger
Dim arrays(length) AsInteger
Dim sortlist(length) AsInteger
Dim usernumber AsInteger
Dim find AsNewfindjob
Dim datesubmitted(length) AsString
Dim numbers AsInteger = 1
Dim nomoreswaps AsBoolean = False
For value = 1To length
            datesubmitted(value) = accounts.datesubmitted(number)
            number = number + 1
Next
For value = 1To length
If SortableDate(datesubmitted(value)) >=
SortableDate(account_sort_certain_period.datefrom.Text) And
SortableDate(datesubmitted(value)) <=
SortableDate(account_sort_certain_period.dateto.Text) Then
                usernumber =
find.findrightaccountdatesubmitteddetails(datesubmitted(value))
                arrays(numbers) = usernumber
                numbers = numbers + 1
                ListBox1.Items.Add(accounts.userid(usernumber) &"        "&
accounts.staffinitials(usernumber) &"                "&
accounts.numberofcopies(usernumber) &"                "&
accounts.datesubmitted(usernumber) &"           £"& accounts.totalprice(usernumber))
EndIf
Next
```

```
If account_sort_certain_period.sortoptions.Text = "Date submitted"Then
For value = 1To length
                datesubmitted(value) = accounts.datesubmitted(array(value))
Next
For value = 1To length

                usernumber =
find.findrightaccountdatesubmitteddetails(datesubmitted(value))
                ListBox1.Items.Add(accounts.userid(usernumber) &"        "&
accounts.staffinitials(usernumber) &"                "&
accounts.numberofcopies(usernumber) &"                "&
accounts.datesubmitted(usernumber) &"            £"& accounts.totalprice(usernumber))
Next
            completed = True
ElseIf account_sort_certain_period.sortoptions.Text = "Number of copies"Then
            number = 0
For value = 1To length
                array(value) = accounts.numberofcopies(arrays(number))
                number = number + 1
Next
Do
                nomoreswaps = True
For value = 1To length
If array(value) > array(value + 1) Then
                        nomoreswaps = False
                        temporystoredvalue = array(value)
                        array(value) = array(value + 1)
                        array(value + 1) = temporystoredvalue
EndIf
Next
LoopUntil nomoreswaps = True
For value = 1To length
'usernumber = find.findrightaccountnumberofcopiesdetails(array(value))
                ListBox1.Items.Add(accounts.userid(array(value)) &"        "&
accounts.staffinitials(array(value)) &"                "&
accounts.numberofcopies(array(value)) &"                "&
accounts.datesubmitted(array(value)) &"            £"&
accounts.totalprice(array(value)))
Next
EndIf
EndSub
Function SortableDate(ByVal Datevalue AsString)
Dim reorderdate AsString
        reorderdate = Datevalue.Substring(8, 2) & Datevalue.Substring(3, 2) &
Datevalue.Substring(0, 2)
        SortableDate = reorderdate
Return SortableDate
EndFunction

PrivateSub reprographicsmenu_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles reprographicsmenu.Click
Reprographics_Accounts.Show()
Me.Close()
EndSub
EndClass
```

Change all or one of the special price(s)

This allows the reprographics to input or all the prices of the special prices if they need to be all changed. The next screen shot shows the windows if you want to just change one of the special price The class 'special price' is used to update the prices on the XML file. (Objective 36)




```
PrivateSub setprices_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles setprices.Click
Dim price AsNewgetnewpricesoneverything
Dim pricelist AsNewsavespecialspricelist

        price.getpapersizea3 = papersizea3.Text
        price.getpapersizea4 = papersizea4.Text
        price.getpapersizea5 = papersizea5.Text
        price.getnormal = normal.Text
        price.getcard = card.Text
        price.getbwprint = bwprint.Text
        price.getcolouredprint = colourprint.Text
        price.getcolourpaper = colouredpaper.Text
        price.getplainpaper = plainpaper.Text
        price.getbacktoback = backtoback.Text
        price.getcollated = collated.Text
        price.getstapled = stapled.Text
        price.getwirobinding = wirobinding.Text
        price.getlaminatea4 = laminatea4.Text
        price.getlaminatea3 = laminatea3.Text
        price.getgluebound = gluebound.Text
        price.gettransparencies = transparencies.Text
        price.gettwoholepunched = twoholepunched.Text
        price.getfolded = folded.Text

        pricelist.Add(price.getpapersizea3, price.getpapersizea4, price.getpapersizea5,
price.getnormal, price.getcard, price.getbwprint, price.getcolouredprint,
price.getcolourpaper, price.getplainpaper, price.getbacktoback, price.getcollated,
price.getstapled, price.getwirobinding, price.getlaminatea4, price.getlaminatea3,
price.getgluebound, price.gettransparencies, price.gettwoholepunched, price.getfolded)
        MsgBox("All the item prices have been successfully set")
Reprographics_Accounts.Show()
Me.Close()
EndSub
```

The get function was used to store the price information in variables which can then be used when using the 'special price list' close to store the information in the XML file:

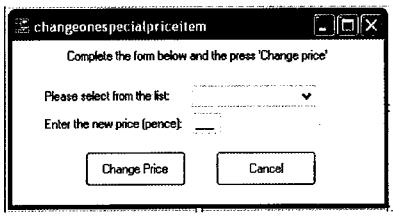
```
PublicClassgetnewpricesoneverything
Private papersizea3 AsInteger
Private papersizea4 AsInteger
Private papersizea5 AsInteger
```

```
Private normal AsInteger
Private card AsInteger
Private bwprint AsInteger
Private colourprint AsInteger
Private plainpaper AsInteger
Private colourpaper AsInteger
Private backtoback AsInteger
Private collated AsInteger
Private stapled AsInteger
Private wirobinding AsInteger
Private laminatea4 AsInteger
Private laminatea3 AsInteger
Private gluebound AsInteger
Private transparencies AsInteger
Private twoholepunched AsInteger
Private folded AsInteger
PublicProperty getpapersizea3 AsInteger
Get
Return papersizea3
EndGet
Set(ByVal value AsInteger)
            papersizea3 = value
EndSet
EndProperty
PublicProperty getpapersizea4 AsInteger
Get
Return papersizea4
EndGet
Set(ByVal value AsInteger)
            papersizea4 = value
EndSet
EndProperty
PublicProperty getpapersizea5 AsInteger
Get
Return papersizea5
EndGet
Set(ByVal value AsInteger)
            papersizea5 = value
EndSet
EndProperty
PublicProperty getnormal AsInteger
Get
Return normal
EndGet
Set(ByVal value AsInteger)
            normal = value
EndSet
EndProperty
PublicProperty getcard AsInteger
Get
Return card
EndGet
Set(ByVal value AsInteger)
            card = value
EndSet
EndProperty
PublicProperty getbwprint AsInteger
Get
Return bwprint
EndGet
Set(ByVal value AsInteger)
            bwprint = value
```

```
EndSet
EndProperty
PublicProperty getcolouredprint AsInteger
Get
Return colourprint
EndGet
Set(ByVal value AsInteger)
            colourprint = value
EndSet
EndProperty
PublicProperty getplainpaper AsInteger
Get
Return plainpaper
EndGet
Set(ByVal value AsInteger)
            plainpaper = value
EndSet
EndProperty
PublicProperty getcolourpaper AsInteger
Get
Return colourpaper
EndGet
Set(ByVal value AsInteger)
            colourpaper = value
EndSet
EndProperty
PublicProperty getbacktoback AsInteger
Get
Return backtoback
EndGet
Set(ByVal value AsInteger)
            backtoback = value
EndSet
EndProperty
PublicProperty getcollated AsInteger
Get
Return collated
EndGet
Set(ByVal value AsInteger)
            collated = value
EndSet
EndProperty
PublicProperty getstapled AsInteger
Get
Return stapled
EndGet
Set(ByVal value AsInteger)
            stapled = value
EndSet
EndProperty
PublicProperty getwirobinding AsInteger
Get
Return wirobinding
EndGet
Set(ByVal value AsInteger)
            wirobinding = value
EndSet
EndProperty
PublicProperty getlaminatea4 AsInteger
Get
Return laminatea4
EndGet
```

Comment [C57]: Stores the information of backtoback to the variable which will be used for example to save the information to the XML file 'Special prices'.

```
Set(ByVal value AsInteger)
            laminatea4 = value
EndSet
EndProperty
PublicProperty getlaminatea3 AsInteger
Get
Return laminatea3
EndGet
Set(ByVal value AsInteger)
            laminatea3 = value
EndSet
EndProperty
PublicProperty getgluebound AsInteger
Get
Return gluebound
EndGet
Set(ByVal value AsInteger)
            gluebound = value
EndSet
EndProperty
PublicProperty gettransparencies AsInteger
Get
Return transparencies
EndGet
Set(ByVal value AsInteger)
            transparencies = value
EndSet
EndProperty
PublicProperty gettwoholepunched AsInteger
Get
Return twoholepunched
EndGet
Set(ByVal value AsInteger)
            twoholepunched = value
EndSet
EndProperty
PublicProperty getfolded AsInteger
Get
Return folded
EndGet
Set(ByVal value AsInteger)
            folded = value
EndSet
EndProperty
EndClass
```



Comment [C58]: By having a screen shot from Windows XP shows that the program can also be run on other operating systems as well.

```
PublicClasschangeonespecialpriceitem

PrivateSub cancel_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles cancel.Click
```

```
Reprographics_Accounts.Show()
Me.Close()
EndSub

PrivateSub changeprice_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles changeprice.Click
Dim getdetails AsNewget_change_price_details
Dim savechanges AsNewspecial_price_list

        getdetails.getitemselected = speicalpriceitemselected.Text
        getdetails.getnewprice = newspecialitemprice.Text
        savechanges.changespecialprice(getdetails.getitemselected,
getdetails.getnewprice)
        MsgBox("Price change successfully")
EndSub
EndClass
```

The code below shows part of the 'special price list' class:

```
Function laminatea3()
Dim specialpriceitem AsNew System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        laminatea3 = priceitems.Item(0).SelectSingleNode("laminatea3").InnerText()
EndFunction
```

Comment [C59]: This function loads the child node named 'laminatea3' and saves the information to the variable 'laminatea3'.

Coding used to create classes

Print job form

```
PublicClassprintjobform
Function jobnumber(ByVal number AsInteger)
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
        jobnumber =
printjobformitems.Item(number).SelectSingleNode("jobnumber").InnerText()
EndFunction
Function userid(ByVal number AsInteger)
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
        userid = printjobformitems.Item(number).SelectSingleNode("userid").InnerText()
EndFunction

Function staffinitials(ByVal number AsInteger)
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
        staffinitials =
printjobformitems.Item(number).SelectSingleNode("staffinitials").InnerText()
EndFunction
```

```
Function requestdate(ByVal number AsInteger)
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
        requestdate =
printjobformitems.Item(number).SelectSingleNode("requestdate").InnerText()
EndFunction
Function copysuppliedas(ByVal number AsInteger)
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
        copysuppliedas =
printjobformitems.Item(number).SelectSingleNode("copysuppliedas").InnerText()
EndFunction
Function papertype(ByVal number AsInteger)
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
        papertype =
printjobformitems.Item(number).SelectSingleNode("papertype").InnerText()
EndFunction
Function papersize(ByVal number AsInteger)
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
        papersize =
printjobformitems.Item(number).SelectSingleNode("papersize").InnerText()
EndFunction
Function numberofcopies(ByVal number AsInteger)
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
        numberofcopies =
printjobformitems.Item(number).SelectSingleNode("numberofcopies").InnerText()
EndFunction
Function papercolour(ByVal number AsInteger)
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
        papercolour =
printjobformitems.Item(number).SelectSingleNode("papercolour").InnerText()
EndFunction
Function backtoback(ByVal number AsInteger)
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
        backtoback =
printjobformitems.Item(number).SelectSingleNode("backtoback").InnerText()
EndFunction
Function collated(ByVal number AsInteger)
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
```

```
        collated =
printjobformitems.Item(number).SelectSingleNode("collated").InnerText()
EndFunction
Function stapled(ByVal number AsInteger)
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
        stapled =
printjobformitems.Item(number).SelectSingleNode("stapled").InnerText()
EndFunction
Function wirobinding(ByVal number AsInteger)
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
        wirobinding =
printjobformitems.Item(number).SelectSingleNode("wirobinding").InnerText()
EndFunction
Function laminate(ByVal number AsInteger)
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
        laminate =
printjobformitems.Item(number).SelectSingleNode("laminate").InnerText()
EndFunction
Function gluebound(ByVal number AsInteger)
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
        gluebound =
printjobformitems.Item(number).SelectSingleNode("gluebound").InnerText()
EndFunction
Function transparencies(ByVal number AsInteger)
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
        transparencies =
printjobformitems.Item(number).SelectSingleNode("transparencies").InnerText()
EndFunction
Function twoholepunched(ByVal number AsInteger)
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
        twoholepunched =
printjobformitems.Item(number).SelectSingleNode("twoholepunched").InnerText()
EndFunction
Function folded(ByVal number AsInteger)
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
        folded = printjobformitems.Item(number).SelectSingleNode("folded").InnerText()
EndFunction
Function proofrequired(ByVal number AsInteger)
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
```

```
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
        proofrequired =
printjobformitems.Item(number).SelectSingleNode("proofrequired").InnerText()
EndFunction
Function jobcomplete(ByVal number AsInteger)
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
        jobcomplete =
printjobformitems.Item(number).SelectSingleNode("completed").InnerText()
EndFunction
Function Length()
Dim jobdoc AsNew System.Xml.XmlDocument
        jobdoc.Load("printjobform.xml")
Dim printjobformitems As System.Xml.XmlNodeList =
jobdoc.SelectNodes("jobs/printjobform")
Return printjobformitems.Count
EndFunction
Function Add(ByVal jobnumber AsInteger, ByVal userid AsInteger, ByVal staffinitials
AsString, ByVal requestdate AsString, ByVal copysuppliedas AsString, ByVal printtype
AsString, ByVal papertype AsString, ByVal papersize AsString, ByVal numberofcopies
AsInteger, ByVal papercolour AsString, ByVal backtoback AsBoolean, ByVal collated
AsBoolean, ByVal stapled AsBoolean, ByVal wirobinding AsBoolean, ByVal laminate
AsBoolean, ByVal gluebound AsBoolean, ByVal transparencies AsBoolean, ByVal
twoholepunched AsBoolean, ByVal folded AsBoolean, ByVal proofrequired AsBoolean)
Dim printjobformxml = XDocument.Load("printjobform.xml")
Dim printjobform AsXElement = _
<printjobform>
<jobnumber><%= jobnumber %></jobnumber>
<userid><%= userid %></userid>
<staffinitials><%= staffinitials %></staffinitials>
<requestdate><%= requestdate %></requestdate>
<copysuppliedas><%= copysuppliedas %></copysuppliedas>
<printtype><%= printtype %></printtype>
<papertype><%= papertype %></papertype>
<papersize><%= papersize %></papersize>
<numberofcopies><%= numberofcopies %></numberofcopies>
<papercolour><%= papercolour %></papercolour>
<backtoback><%= backtoback %></backtoback>
<collated><%= collated %></collated>
<stapled><%= stapled %></stapled>
<wirobinding><%= wirobinding %></wirobinding>
<laminate><%= laminate %></laminate>
<gluebound><%= gluebound %></gluebound>
<transparencies><%= transparencies %></transparencies>
<twoholepunched><%= twoholepunched %></twoholepunched>
<folded><%= folded %></folded>
<proofrequired><%= proofrequired %></proofrequired>
<completed>false</completed>
</printjobform>
        printjobformxml.<jobs>(0).Add(printjobform)
        printjobformxml.Save("printjobform.xml")
ReturnTrue
EndFunction
Sub jobcompleted(ByVal job AsString)

Dim printjobformxml = XDocument.Load("printjobform.xml")
Dim findjob AsNewfindjob
Dim jobnumber AsInteger
        jobnumber = findjob.findrightjob(job)
```

Comment [C60]: Saves information to the XML file 'Print job form'.

```
printjobformxml.<jobs>(0).<printjobform>(jobnumber).<completed>(0).SetValue("true")
        printjobformxml.Save("printjobform.xml")
EndSub
EndClass
```

Print credits

```
PublicClassprintcredits
Function userid(ByVal number AsInteger)
Dim message AsNew System.Xml.XmlDocument
        message.Load("printcredits.xml")
Dim printcreditdetails As System.Xml.XmlNodeList =
message.SelectNodes("userprintcreditprofile/printcredit")
        userid = printcreditdetails.Item(number).SelectSingleNode("userid").InnerText()
EndFunction
Function printcredits(ByVal number AsInteger)
Dim message AsNew System.Xml.XmlDocument
        message.Load("printcredits.xml")
Dim printcreditdetails As System.Xml.XmlNodeList =
message.SelectNodes("userprintcreditprofile/printcredit")
        printcredits =
printcreditdetails.Item(number).SelectSingleNode("printcredits").InnerText()
EndFunction
Function Length()
Dim message AsNew System.Xml.XmlDocument
        message.Load("printcredits.xml")
Dim printcreditdetails As System.Xml.XmlNodeList =
message.SelectNodes("userprintcreditprofile/printcredit")
Return printcreditdetails.Count
EndFunction
Function saveprintcredits(ByVal userid AsInteger, ByVal printcredits AsInteger)
Dim printcreditdetailsxml = XDocument.Load("printcredits.xml")
Dim printcreditdetail AsXElement = _
<printcredit>
<userid>[redacted] userid [redacted]</userid>
<printcredits>[redacted] printcredits [redacted]</printcredits>
</printcredit>

        printcreditdetailsxml.<userprintcreditprofile>(0).Add(printcreditdetail)
        printcreditdetailsxml.Save("printcredits.xml")
ReturnTrue
EndFunction
Function updateprintcredits(ByVal userid AsInteger, ByVal value AsInteger)
Dim checkprintcredits AsNewprintcredits
Dim currentprintcreditnumber AsInteger
Dim updatedprintcreditnumber AsInteger
Dim profilenumber AsInteger
Dim findprofilenumber AsNewfinduser

        profilenumber = findprofilenumber.finduserprofile(userid)
        currentprintcreditnumber = checkprintcredits.printcredits(profilenumber)
        updatedprintcreditnumber = currentprintcreditnumber - value
If updatedprintcreditnumber <= 0Then
            updateprintcredits = False
Else
            updateprintcredits = True
            changeprintcredits(profilenumber, updatedprintcreditnumber)
EndIf
Return updateprintcredits
EndFunction
Sub changeprintcredits(ByVal profilenumber AsInteger, ByVal newvalue AsInteger)
```

```
Dim printcreditxml = XDocument.Load("printcredits.xml")

printcreditxml.<userprintcreditprofile>(0).<printcredit>(profilenumber).<printcredits>(
0).SetValue(newvalue)
        printcreditxml.Save("printcredits.xml")
EndSub

Function lookupprintcredits(ByVal number AsInteger)
Dim printcreditdetails AsNew System.Xml.XmlDocument
        printcreditdetails.Load("printcredits.xml")
Dim printcreditdetail As System.Xml.XmlNodeList =
printcreditdetails.SelectNodes("userprintcreditprofile/printcredit")
        lookupprintcredits =
printcreditdetail.Item(number).SelectSingleNode("printcredits").InnerText()
EndFunction
EndClass
```

Special price list

```
PublicClassspecial_price_list
Function papersizea3()
Dim specialpriceitem AsNew System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        papersizea3 = speicalpricedetails.Item(0).SelectSingleNode("a3").InnerText()
EndFunction
Function papersizea4()
Dim specialpriceitem AsNew System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        papersizea4 = speicalpricedetails.Item(0).SelectSingleNode("a4").InnerText()
EndFunction
Function papersizea5()
Dim specialpriceitem AsNew System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        papersizea5 = speicalpricedetails.Item(0).SelectSingleNode("a5").InnerText()
EndFunction
Function normal()
Dim specialpriceitem AsNew System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        normal = speicalpricedetails.Item(0).SelectSingleNode("normal").InnerText()
EndFunction
Function card()
Dim specialpriceitem AsNew System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        card = speicalpricedetails.Item(0).SelectSingleNode("card").InnerText()
EndFunction
Function papercolour()
Dim specialpriceitem AsNew System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        papercolour =
speicalpricedetails.Item(0).SelectSingleNode("papercolour").InnerText()
```

```
EndFunction
Function bwprint()
Dim specialpriceitem AsNew System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        bwprint =
speicalpricedetails.Item(0).SelectSingleNode("blackandwhite").InnerText()
EndFunction
Function colourprint()
Dim specialpriceitem AsNew System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        colourprint =
speicalpricedetails.Item(0).SelectSingleNode("colour").InnerText()
EndFunction
Function plainpaper()
Dim specialpriceitem AsNew System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        plainpaper =
speicalpricedetails.Item(0).SelectSingleNode("blackandwhite").InnerText()
EndFunction
Function backtoback()
Dim specialpriceitem AsNew System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        backtoback =
speicalpricedetails.Item(0).SelectSingleNode("backtoback").InnerText()
EndFunction
Function collated()
Dim specialpriceitem AsNew System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        collated = speicalpricedetails.Item(0).SelectSingleNode("collated").InnerText()
EndFunction
Function stapled()
Dim specialpriceitem AsNew System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        stapled = speicalpricedetails.Item(0).SelectSingleNode("stapled").InnerText()
EndFunction
Function wirobinding()
Dim specialpriceitem AsNew System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        wirobinding =
speicalpricedetails.Item(0).SelectSingleNode("wirobinding").InnerText()
EndFunction
Function laminatea4()
Dim specialpriceitem AsNew System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        laminatea4 =
speicalpricedetails.Item(0).SelectSingleNode("laminatea4").InnerText()
```

```
EndFunction
Function laminatea3()
Dim specialpriceitem AsNew System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        laminatea3 =
speicalpricedetails.Item(0).SelectSingleNode("laminatea3").InnerText()
EndFunction
Function gluebound()
Dim specialpriceitem AsNew System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        gluebound =
speicalpricedetails.Item(0).SelectSingleNode("gluebound").InnerText()
EndFunction
Function transparencies()
Dim specialpriceitem AsNew System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        transparencies =
speicalpricedetails.Item(0).SelectSingleNode("transparencies").InnerText()
EndFunction
Function twoholepunched()
Dim specialpriceitem AsNew System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        twoholepunched =
speicalpricedetails.Item(0).SelectSingleNode("twoholepunched").InnerText()
EndFunction
Function folded()
Dim specialpriceitem AsNew System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        folded = speicalpricedetails.Item(0).SelectSingleNode("folded").InnerText()
EndFunction
Function Add(ByVal a3 AsString, ByVal a4 AsString, ByVal a5 AsString, ByVal normal
AsString, ByVal card AsString, ByVal bwprint AsString, ByVal colourprint AsString,
ByVal papercolour AsString, ByVal plainpaper AsString, ByVal backtoback AsString, ByVal
collated AsString, ByVal stapled AsString, ByVal wirobinding AsString, ByVal laminatea4
AsString, ByVal laminatea3 AsString, ByVal gluebound AsString, ByVal transparencies
AsString, ByVal twoholepunched AsString, ByVal folded AsString)
Dim specialpricexml = XDocument.Load("specialprices.xml")
        specialpricexml.<prices>(0).RemoveNodes()
Dim specialprice AsXElement = _
<specialprice>
<a3><%= a3 %></a3>
<a4><%= a4 %></a4>
<a5><%= a5 %></a5>
<normal><%= normal %></normal>
<card><%= card %></card>
<blackandwhite><%= bwprint %></blackandwhite>
<colour><%= colourprint %></colour>
<papercolour><%= papercolour %></papercolour>
<paperbandw><%= plainpaper %></paperbandw>
<backtoback><%= backtoback %></backtoback>
<collated><%= collated %></collated>
<stapled><%= stapled %></stapled>
```

```
<wirobinding><%= wirobinding %></wirobinding>
<laminatea4><%= laminatea4 %></laminatea4>
<laminatea3><%= laminatea3 %></laminatea3>
<gluebound><%= gluebound %></gluebound>
<transparencies><%= transparencies %></transparencies>
<twoholepunched><%= twoholepunched %></twoholepunched>
<folded><%= folded %></folded>
</specialprice>
        specialpricexml.<prices>(0).Add(specialprice)
        specialpricexml.Save("specialprices.xml")
ReturnTrue
EndFunction
Sub changespecialprice(ByVal item AsString, ByVal newprice AsString)
If item = "a3"Then
            a3(item, newprice)
ElseIf item = "a4"Then
            a4(item, newprice)
ElseIf item = "a5"Then
            a5(item, newprice)
ElseIf item = "normal"Then
            normal(item, newprice)
ElseIf item = "card"Then
            card(item, newprice)
ElseIf item = "blackandwhite"Then
            blackandwhite(item, newprice)
ElseIf item = "colour"Then
            colour(item, newprice)
ElseIf item = "papercolour"Then
            papercolour(item, newprice)
ElseIf item = "paperbandw"Then
            paperbandw(item, newprice)
ElseIf item = "backtoback"Then
            backtoback(item, newprice)
ElseIf item = "collated"Then
            collated(item, newprice)
ElseIf item = "stapled"Then
            stapled(item, newprice)
ElseIf item = "wirobinding"Then
            wirobinding(item, newprice)
ElseIf item = "laminatea4"Then
            laminatea4(item, newprice)
ElseIf item = "laminatea3"Then
            laminatea3(item, newprice)
ElseIf item = "gluebound"Then
            gluebound(item, newprice)
ElseIf item = "transparencies"Then
            transparencies(item, newprice)
ElseIf item = "twoholepunched"Then
            twoholepunched(item, newprice)
ElseIf item = "folded"Then
            folded(item, newprice)
EndIf
EndSub
Sub a3(ByVal item AsString, ByVal newprice AsString)
Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<a3>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
EndSub
Sub a4(ByVal item AsString, ByVal newprice AsString)
Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<a4>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
```

Comment [C61]: Saves the information to the XML file 'special prices'

```
EndSub
Sub a5(ByVal item AsString, ByVal newprice AsString)
Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<a5>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
EndSub
Sub normal(ByVal item AsString, ByVal newprice AsString)
Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<normal>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
EndSub
Sub card(ByVal item AsString, ByVal newprice AsString)
Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<card>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
EndSub
Sub blackandwhite(ByVal item AsString, ByVal newprice AsString)
Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<blackandwhite>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
EndSub
Sub colour(ByVal item AsString, ByVal newprice AsString)
Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<colour>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
EndSub
Sub papercolour(ByVal item AsString, ByVal newprice AsString)
Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<papercolour>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
EndSub
Sub paperbandw(ByVal item AsString, ByVal newprice AsString)
Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<paperbandw>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
EndSub
Sub backtoback(ByVal item AsString, ByVal newprice AsString)
Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<backtoback>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
EndSub
Sub collated(ByVal item AsString, ByVal newprice AsString)
Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<collated>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
EndSub
Sub stapled(ByVal item AsString, ByVal newprice AsString)
Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<stapled>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
EndSub
Sub wirobinding(ByVal item AsString, ByVal newprice AsString)
Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<wirobinding>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
EndSub
Sub laminatea4(ByVal item AsString, ByVal newprice AsString)
Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<laminatea4>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
EndSub
Sub laminatea3(ByVal item AsString, ByVal newprice AsString)
```

Comment [C62]: Changes the price of the special price item in the 'special prices' XML file.

```
Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<laminatea3>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
EndSub
Sub gluebound(ByVal item AsString, ByVal newprice AsString)
Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<gluebound>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
EndSub
Sub transparencies(ByVal item AsString, ByVal newprice AsString)
Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<transparencies>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
EndSub
Sub twoholepunched(ByVal item AsString, ByVal newprice AsString)
Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<twoholepunched>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
EndSub
Sub folded(ByVal item AsString, ByVal newprice AsString)
Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<folded>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
EndSub
EndClass
```

User profile

```
PublicClassuserprofile
Function userid(ByVal number AsInteger)
Dim userdetails AsNew System.Xml.XmlDocument
        userdetails.Load("userdetails.xml")
Dim userdetail As System.Xml.XmlNodeList = userdetails.SelectNodes("users/user")
        userid = userdetail.Item(number).SelectSingleNode("userid").InnerText()
EndFunction
Function department(ByVal number AsInteger)
Dim userdetails AsNew System.Xml.XmlDocument
        userdetails.Load("userdetails.xml")
Dim userdetail As System.Xml.XmlNodeList = userdetails.SelectNodes("users/user")
        department = userdetail.Item(number).SelectSingleNode("department").InnerText()
EndFunction
Function password(ByVal number AsInteger)
Dim userdetails AsNew System.Xml.XmlDocument
        userdetails.Load("userdetails.xml")
Dim userdetail As System.Xml.XmlNodeList = userdetails.SelectNodes("users/user")
        password = userdetail.Item(number).SelectSingleNode("password").InnerText()
EndFunction
Function securityquestion(ByVal number AsInteger)
Dim userdetails AsNew System.Xml.XmlDocument
        userdetails.Load("userdetails.xml")
Dim userdetail As System.Xml.XmlNodeList = userdetails.SelectNodes("users/user")
        securityquestion =
userdetail.Item(number).SelectSingleNode("securityquestion").InnerText()
EndFunction
Function securityanswer(ByVal number AsInteger)
Dim userdetails AsNew System.Xml.XmlDocument
        userdetails.Load("userdetails.xml")
Dim userdetail As System.Xml.XmlNodeList = userdetails.SelectNodes("users/user")
        securityanswer =
userdetail.Item(number).SelectSingleNode("securityanswer").InnerText()
EndFunction
```

```
Function Length()
Dim userdetails AsNew System.Xml.XmlDocument
        userdetails.Load("userdetails.xml")
Dim userdetail As System.Xml.XmlNodeList = userdetails.SelectNodes("users/user")
Return userdetail.Count
EndFunction
Sub changepassword(ByVal number AsInteger, ByVal newpassword AsString)
Dim userxml = XDocument.Load("userdetails.xml")
        userxml.<users>(0).<user>(number).<password>(0).SetValue(newpassword)
        userxml.<users>(0).<user>(number).<confirmpassword>(0).SetValue(newpassword)
        userxml.Save("userdetails.xml")
EndSub
Sub saveuser(ByVal userid AsInteger, ByVal department AsString, ByVal password
AsString, ByVal confirmpassword AsString, ByVal securityquestion AsString, ByVal
securityanswer AsString)
Dim userdetails = XDocument.Load("userdetails.xml")
Dim user AsXElement = _
<user>
<userid>[redacted] userid [redacted]</userid>
<department>[redacted] department [redacted]</department>
<password>[redacted] password [redacted]</password>
<confirmpassword>[redacted] confirmpassword [redacted]</confirmpassword>
<securityquestion>[redacted] securityquestion [redacted]</securityquestion>
<securityanswer>[redacted] securityanswer [redacted]</securityanswer>
</user>
        userdetails.<users>(0).Add(user)
        userdetails.Save("userdetails.xml")
EndSub
EndClass
```

User sent message

```
PublicClassuser_sent_message
Function addmessage(ByVal messagenumber AsInteger, ByVal userid AsInteger, ByVal name
AsString, ByVal datesubmitted AsString, ByVal topic AsString, ByVal message AsString)
Dim messagedetailsxml = XDocument.Load("messages.xml")
Dim messagedetail AsXElement = _
<message>
<messagenumber><%= messagenumber %></messagenumber>
<userid><%= userid %></userid>
<name><%= name %></name>
<datesubmitted><%= datesubmitted %></datesubmitted>
<topic><%= topic %></topic>
<message><%= message %></message>
<read><%=False%></read>
</message>
        messagedetailsxml.<messages>(0).Add(messagedetail)
        messagedetailsxml.Save("messages.xml")
ReturnTrue
EndFunction
Function messagenumber(ByVal number AsInteger)
Dim message AsNew System.Xml.XmlDocument
        message.Load("messages.xml")
Dim messagedetails As System.Xml.XmlNodeList = message.SelectNodes("messages/message")
        messagenumber =
messagedetails.Item(number).SelectSingleNode("messagenumber").InnerText()
EndFunction
Function userid(ByVal number AsInteger)
Dim message AsNew System.Xml.XmlDocument
        message.Load("messages.xml")
Dim messagedetails As System.Xml.XmlNodeList = message.SelectNodes("messages/message")
```

Comment [C63]: Saves information in the XML file.

```
        userid = messagedetails.Item(number).SelectSingleNode("userid").InnerText()
EndFunction
Function name(ByVal number AsInteger)
Dim message AsNew System.Xml.XmlDocument
        message.Load("messages.xml")
Dim messagedetails As System.Xml.XmlNodeList = message.SelectNodes("messages/message")
        name = messagedetails.Item(number).SelectSingleNode("name").InnerText()
EndFunction
Function datesubmitted(ByVal number AsInteger)
Dim message AsNew System.Xml.XmlDocument
        message.Load("messages.xml")
Dim messagedetails As System.Xml.XmlNodeList = message.SelectNodes("messages/message")
        datesubmitted =
messagedetails.Item(number).SelectSingleNode("datesubmitted").InnerText()
EndFunction
Function topic(ByVal number AsInteger)
Dim message AsNew System.Xml.XmlDocument
        message.Load("messages.xml")
Dim messagedetails As System.Xml.XmlNodeList = message.SelectNodes("messages/message")
        topic = messagedetails.Item(number).SelectSingleNode("topic").InnerText()
EndFunction
Function submittedmessage(ByVal number AsInteger)
Dim message AsNew System.Xml.XmlDocument
        message.Load("messages.xml")
Dim messagedetails As System.Xml.XmlNodeList = message.SelectNodes("messages/message")
        submittedmessage =
messagedetails.Item(number).SelectSingleNode("message").InnerText()
EndFunction
Function read(ByVal number AsInteger)
Dim message AsNew System.Xml.XmlDocument
        message.Load("messages.xml")
Dim messagedetails As System.Xml.XmlNodeList = message.SelectNodes("messages/message")
        read = messagedetails.Item(number).SelectSingleNode("read").InnerText()
EndFunction
Function length()
Dim message AsNew System.Xml.XmlDocument
        message.Load("messages.xml")
Dim messagedetails As System.Xml.XmlNodeList = message.SelectNodes("messages/message")
Return messagedetails.Count
EndFunction
Sub setmessagetoread(ByVal job AsString)
Dim messagexml = XDocument.Load("messages.xml")
Dim message AsNewfindjob
Dim messagenumber AsInteger
        messagenumber = message.findrightmessage(job)
        messagexml.<messages>(0).<message>(messagenumber).<read>(0).SetValue("true")
        messagexml.Save("messages.xml")
EndSub
EndClass
```

Reprographics sent message

```
PublicClassreprographics_sent_messages
Function messagenumber(ByVal number AsInteger)
Dim message AsNew System.Xml.XmlDocument
        message.Load("reprographicssentmessages.xml")
Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("reprographicssentmessages/message")
        messagenumber =
messagedetails.Item(number).SelectSingleNode("messagenumber").InnerText()
EndFunction
Function userid(ByVal number AsInteger)
```

```
Dim message AsNew System.Xml.XmlDocument
        message.Load("reprographicssentmessages.xml")
Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("reprographicssentmessages/message")
        userid = messagedetails.Item(number).SelectSingleNode("userid").InnerText()
EndFunction
Function name(ByVal number AsInteger)
Dim message AsNew System.Xml.XmlDocument
        message.Load("reprographicssentmessages.xml")
Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("reprographicssentmessages/message")
        name = messagedetails.Item(number).SelectSingleNode("name").InnerText()
EndFunction
Function datesubmitted(ByVal number AsInteger)
Dim message AsNew System.Xml.XmlDocument
        message.Load("reprographicssentmessages.xml")
Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("reprographicssentmessages/message")
        datesubmitted =
messagedetails.Item(number).SelectSingleNode("datesent").InnerText()
EndFunction
Function topic(ByVal number AsInteger)
Dim message AsNew System.Xml.XmlDocument
        message.Load("reprographicssentmessages.xml")
Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("reprographicssentmessages/message")
        topic = messagedetails.Item(number).SelectSingleNode("topic").InnerText()
EndFunction
Function submittedmessage(ByVal number AsInteger)
Dim message AsNew System.Xml.XmlDocument
        message.Load("reprographicssentmessages.xml")
Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("reprographicssentmessages/message")
        submittedmessage =
messagedetails.Item(number).SelectSingleNode("message").InnerText()
EndFunction
Function read(ByVal number AsInteger)
Dim message AsNew System.Xml.XmlDocument
        message.Load("reprographicssentmessages.xml")
Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("reprographicssentmessages/message")
        read = messagedetails.Item(number).SelectSingleNode("messageread").InnerText()
EndFunction
Sub setmessagetoread(ByVal message AsString)
Dim messagexml = XDocument.Load("reprographicssentmessages.xml")
Dim findmessageprofile AsNewfindjob
Dim messagenumber AsInteger
        messagenumber = findmessageprofile.findrightreprographicsmessage(message)

messagexml.<reprographicssentmessages>(0).<message>(messagenumber).<messageread>(0).Set
Value("true")
        messagexml.Save("reprographicssentmessages.xml")
EndSub
Function Length()
Dim message AsNew System.Xml.XmlDocument
        message.Load("reprographicssentmessages.xml")
Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("reprographicssentmessages/message")
Return messagedetails.Count()
EndFunction
```

```
Function saveregraphicsentmessage(ByVal messagenumber AsInteger, ByVal userid
AsInteger, ByVal name AsString, ByVal datesent AsString, ByVal topic AsString, ByVal
message AsString)
Dim reprographicsmessagesentdetailsxml =
XDocument.Load("reprographicssentmessages.xml")
Dim messagesentdetail AsXElement = _
<message>
<messagenumber><%= messagenumber %></messagenumber>
<userid><%= userid %></userid>
<name><%= name %></name>
<datesent><%= datesent %></datesent>
<topic><%= topic %></topic>
<message><%= message %></message>
<messageread><%=False%></messageread>
</message>

reprographicsmessagesentdetailsxml.<reprographicssentmessages>(0).Add(messagesentdetail
)
        reprographicsmessagesentdetailsxml.Save("reprographicssentmessages.xml")
ReturnTrue
EndFunction
EndClass
```

Comment [C64]: Saves the information on the XML file.

Queue and stack

```
PublicClassaddtoqueue
    Function addformtoqueue()
        Print_job_queue.Queue.Items.Clear()
        Dim jobitem As New printjobform
        Dim length As Integer
        Dim topofstack As String
        Dim number As Integer = 0
        Dim currentnumber As Integer = 0
        Dim nomoreswaps As Boolean
        Dim temporystoredvalue As String
        Dim childnumber As Integer
        Dim find As New findjob
        Dim initialisenumber As Integer
        length = jobitem.Length
        Dim sort(length) As String
        For initialisenumber = 1 To length
            sort(initialisenumber) = ""
        Next
        For number = 1 To length
            If jobitem.jobcomplete(number - 1) = "false" Then
                currentnumber = currentnumber + 1
                sort(currentnumber) = jobitem.requestdate(number - 1)
            End If
        Next
        Do
            nomoreswaps = True
            For value = 1 To length - 1
                If sort(value + 1) <> "" Then
                    If SortableDate(sort(value)) > SortableDate(sort(value + 1)) Then
                        nomoreswaps = False
                        temporystoredvalue = sort(value)
                        sort(value) = sort(value + 1)
                        sort(value + 1) = temporystoredvalue
                    End If
                End If
            Next
```

```
        Loop Until nomoreswaps = True
        For value = 1 To length
            If sort(value) <> "" Then
                childnumber = find.findrequestdate(sort(value))
                Print_job_queue.Queue.Items.Add(jobitem.jobnumber(childnumber) & "  " &
jobitem.requestdate(childnumber) & "  " & jobitem.staffinitials(childnumber))
            End If
        Next
        Try
            topofstack = Print_job_queue.Queue.Items.Item(0)
        Catch
            topofstack = 0
            MsgBox("no items in the queue", MsgBoxStyle.OkOnly, "No Jobs")
        End Try
        Return topofstack
    End Function

Sub addaccountdetails()
Dim accounts AsNewaccounts
Dim length AsInteger
Dim number AsInteger = 0
        length = accounts.Length
        Reprographics_Accounts.ListBox1.Items.Clear()
For number = 1To length
            Reprographics_Accounts.ListBox1.Items.Add(accounts.userid(number - 1) &"
"& accounts.staffinitials(number - 1) &"                "&
accounts.numberofcopies(number - 1) &"            "&"            "&
accounts.datesubmitted(number - 1) &"           £"& accounts.totalprice(number - 1))
Next
EndSub
Sub addnewmessagetostack()
Dim messageitem AsNewuser_sent_message
Dim length AsInteger
Dim number AsInteger
        reprographics_check_messages.newmessagestack.Items.Clear()
        length = messageitem.length
For number = 1To length
If messageitem.read(number - 1) = "false"Then

reprographics_check_messages.newmessagestack.Items.Add(messageitem.messagenumber(number
- 1) &"   "& messageitem.userid(number - 1) &"      "& messageitem.name(number - 1) &"
"& messageitem.datesubmitted(number - 1) &"      "& messageitem.topic(number - 1))
EndIf
Next

EndSub
Sub addpastmessagetoqueue()
Dim messageitem AsNewuser_sent_message
Dim length AsInteger
Dim number AsInteger = 0
        length = messageitem.length
        Reprographics_past_messages.pastmessagesstack.Items.Clear()
For number = 1To length
If messageitem.read(number - 1) = "true"Then

Reprographics_past_messages.pastmessagesstack.Items.Add(messageitem.messagenumber(numbe
r - 1) &"   "& messageitem.userid(number - 1) &"      "& messageitem.name(number - 1)
&"      "& messageitem.datesubmitted(number - 1) &"      "& messageitem.topic(number -
1))
EndIf
Next
EndSub
```

Comment [C65]: Bubblesort procedure to sort data.

Comment [C66]: This is used to catch the error if 'top of stack' equals a null function.

```
Sub addtolistpastjobs()
Dim submittedjobs AsNewprintjobform
Dim length AsInteger
Dim number AsInteger = 0
        length = submittedjobs.Length
        View_print_forms.ListBox1.Items.Clear()
For number = 1To length
If submittedjobs.userid(number - 1) = Login.username.Text Then
                View_print_forms.ListBox1.Items.Add(submittedjobs.jobnumber(number - 1)
&"  "& submittedjobs.requestdate(number - 1) &"  "& submittedjobs.staffinitials(number
- 1))
EndIf
Next
EndSub
Sub adduserpastmessagetostack()
Dim messageitem AsNewreprographics_sent_messages
Dim findusernumber AsNewfinduser
Dim length AsInteger
Dim usernumber AsInteger
Dim number AsInteger = 0
        usercheckpastmessages.messagepastmessagestack.Items.Clear()
        length = messageitem.Length
        usernumber =
findusernumber.finduserprofileformessagesentbyreprographics(Login.username.Text)
For number = 1To length
If messageitem.read(number - 1) = "true"And Login.username.Text =
messageitem.userid(number - 1) Then

usercheckpastmessages.messagepastmessagestack.Items.Add(messageitem.messagenumber(numbe
r - 1) &"   "& messageitem.userid(number - 1) &"      "& messageitem.name(number - 1)
&"      "& messageitem.datesubmitted(number - 1) &"      "& messageitem.topic(number -
1))
EndIf
Next
EndSub
Sub addusernewmessagetoqueue(ByVal value)
Dim messageitem AsNewreprographics_sent_messages
Dim findusernumber AsNewfinduser
Dim doesuserhaveanewmessage AsNewdoes_user_have_a_new_message
Dim length AsInteger
Dim usernumber AsInteger
Dim newmessage AsBoolean
        length = messageitem.Length
        userchecknewmessages.messagestack.Items.Clear()
        newmessage = doesuserhaveanewmessage.userprofilefound(value)
If newmessage = TrueThen
           usernumber =
findusernumber.finduserprofileformessagesentbyreprographics(value)
For number = 1To length
If messageitem.read(number - 1) = "false"And Login.username.Text =
messageitem.userid(number - 1) Then

usercheckpastmessages.messagestack.Items.Add(messageitem.messagenumber(number - 1) &"
"& messageitem.userid(number - 1) &"      "& messageitem.name(number - 1) &"      "&
messageitem.datesubmitted(number - 1) &"      "& messageitem.topic(number - 1))
EndIf
Next
EndIf
EndSub
EndClass
```

Comment [C67]: Initializes, clears, adds, and shows the items in the stack. The functions above do the similar procedure. This is true for all the functions an procedures in the Queue class.

Find user

```
PublicClassfinduser
Function findrightuser(ByVal username AsInteger, ByVal password AsString)
Dim usernumber AsNewuserprofile
Dim length AsInteger
Dim number AsInteger
Dim validuser AsBoolean
Dim foundusernumber AsInteger
Dim loginvalid AsBoolean
Dim a AsString
        length = usernumber.Length
        a = username & password
        validuser = userfound(a)
Do
If a = (usernumber.userid(number) & usernumber.password(number)) Then
                loginvalid = True
                foundusernumber = number

Else
                loginvalid = False
                number = number + 1
EndIf
LoopUntil number = length Or loginvalid = True

        findrightuser = foundusernumber + 1
Return findrightuser
EndFunction

Function userpasswordfound(ByVal value)
Dim number AsInteger = 0
Dim checkuserdata AsNewuserprofile

Do
If value = (checkuserdata.userid(number) & checkuserdata.password(number)) Then
                userpasswordfound = True
Else
                userpasswordfound = False
                number = number + 1
EndIf
LoopUntil number = checkuserdata.Length Or userpasswordfound = True
Return userpasswordfound
EndFunction
Function findusersequrityquestion(ByVal userid AsString)
Dim usernumber AsNewuserprofile
Dim length AsInteger
Dim number AsInteger
Dim foundusernumber AsInteger
Dim loginvalid AsBoolean
Dim a AsString
        a = userid
        length = usernumber.Length
        number = 0
Do
If a = usernumber.userid(number) Then
                loginvalid = True
                foundusernumber = number

Else
                loginvalid = False
                number = number + 1
```

Comment [C68]: Used to check the information submitted on the login page is valid or not. Uses the 'user profile class to see if the details entered on the login page match what is in the XML file

```
EndIf
LoopUntil number = length Or loginvalid = True

        findusersequrityquestion = foundusernumber
Return findusersequrityquestion
EndFunction

Function userfound(ByVal value)
Dim number AsInteger
Dim checkuserdata AsNewuserprofile
        number = 0
Do
If value = (checkuserdata.userid(number)) Then
                userfound = True
Else
                userfound = False
                number = number + 1
EndIf
LoopUntil number = checkuserdata.Length Or userfound = True
Return userfound
EndFunction
Function finduserprofile(ByVal value)
Dim number AsInteger
Dim checkprofiledata AsNewuserprofile
Dim profilefound AsBoolean
        number = 0
Do
If value = (checkprofiledata.userid(number)) Then
                profilefound = True
Else
                profilefound = False
                number = number + 1
EndIf
LoopUntil number = checkprofiledata.Length - 1Or profilefound = True
Return number
EndFunction
Function finduserprofileformessagesentbyreprographics(ByVal value)
Dim number AsInteger
Dim checkprofiledata AsNewreprographics_sent_messages
Dim profilefound AsBoolean
        number = 0
Do
If value = (checkprofiledata.userid(number)) Then
                profilefound = True
Else
                profilefound = False
                number = number + 1
EndIf
LoopUntil number = checkprofiledata.Length Or profilefound = True
Return number
EndFunction
Function finddepartment(ByVal value) AsString
Dim checkprofiledata AsNewuserprofile
        finddepartment = checkprofiledata.department(value)
EndFunction
EndClass
```

Comment [C69]: This function uses the 'userprofile' class to find the security question. This function is used when the user has forgotten their password, go through the security checks.

XML files

Accountdetails.xml

```
<?xmlversion="1.0"encoding="utf-8"?>
<accounts>
<useraccountdetails>
<jobnumber>1</jobnumber>
<userid>2204</userid>
<staffinitials>abc</staffinitials>
<department>Art</department>
<datesubmitted>02/04/2010</datesubmitted>
<numberofcopies>4</numberofcopies>
<totalprice>1616</totalprice>
</useraccountdetails>
<useraccountdetails>
<jobnumber>2</jobnumber>
<userid>2204</userid>
<staffinitials>abc</staffinitials>
<department>Art</department>
<datesubmitted>25/03/2010</datesubmitted>
<numberofcopies>7</numberofcopies>
<totalprice>28</totalprice>
</useraccountdetails>
<useraccountdetails>
<jobnumber>3</jobnumber>
<userid>2205</userid>
<staffinitials>rwa</staffinitials>
<department>Biology</department>
<datesubmitted>04/04/2010</datesubmitted>
<numberofcopies>5</numberofcopies>
<totalprice>220</totalprice>
</useraccountdetails>
<useraccountdetails>
<jobnumber>4</jobnumber>
<userid>2205</userid>
<staffinitials>rwa</staffinitials>
<department>Biology</department>
<datesubmitted>10/04/2010</datesubmitted>
<numberofcopies>4</numberofcopies>
<totalprice>16</totalprice>
</useraccountdetails>
</accounts>
```

Comment [C70]: Account profile which is based on the print job form that has been submitted

Messages.xml

```
<?xmlversion="1.0"encoding="utf-8"?>
<messages>
<message>
<messagenumber>1</messagenumber>
<userid>2204</userid>
<name>abc</name>
<datesubmitted>03/02/2010</datesubmitted>
<topic>Message test</topic>
<message>Reply if you have read this message.</message>
<read>true</read>
</message>
</messages>
```

Printcredits.xml

```
<?xmlversion="1.0"encoding="utf-8"?>
<userprintcreditprofile>
<printcredit>
```

```
<userid>8000</userid>
<printcredits>1000</printcredits>
</printcredit>
<printcredit>
<userid>2204</userid>
<printcredits>989</printcredits>
</printcredit>
<printcredit>
<userid>2205</userid>
<printcredits>991</printcredits>
</printcredit>
</userprintcreditprofile>
```

Comment [C71]: Print credit profile

Printjobform.xml

```
<?xmlversion="1.0"encoding="utf-8"?>
<jobs>
<printjobform>
<jobnumber>1</jobnumber>
<userid>2204</userid>
<staffinitials>abc</staffinitials>
<requestdate>02/04/2010</requestdate>
<copysuppliedas>CD-ROM</copysuppliedas>
<printtype>Colour</printtype>
<papertype>Normal</papertype>
<papersize>A4</papersize>
<numberofcopies>4</numberofcopies>
<papercolour>Blue</papercolour>
<backtoback>false</backtoback>
<collated>true</collated>
<stapled>false</stapled>
<wirobinding>false</wirobinding>
<laminate>false</laminate>
<gluebound>true</gluebound>
<transparencies>true</transparencies>
<twoholepunched>false</twoholepunched>
<folded>false</folded>
<proofrequired>false</proofrequired>
<completed>true</completed>
</printjobform>
<printjobform>
<jobnumber>2</jobnumber>
<userid>2204</userid>
<staffinitials>abc</staffinitials>
<requestdate>25/03/2010</requestdate>
<copysuppliedas>HARD Copy</copysuppliedas>
<printtype>Black and White</printtype>
<papertype>Normal</papertype>
<papersize>A4</papersize>
<numberofcopies>7</numberofcopies>
<papercolour>Green</papercolour>
<backtoback>true</backtoback>
<collated>true</collated>
<stapled>false</stapled>
<wirobinding>false</wirobinding>
<laminate>false</laminate>
<gluebound>true</gluebound>
<transparencies>false</transparencies>
<twoholepunched>false</twoholepunched>
<folded>false</folded>
<proofrequired>true</proofrequired>
```

```
<completed>false</completed>
</printjobform>
<printjobform>
<jobnumber>3</jobnumber>
<userid>2205</userid>
<staffinitials>rwa</staffinitials>
<requestdate>04/04/2010</requestdate>
<copysuppliedas>CD-ROM</copysuppliedas>
<printtype>Colour</printtype>
<papertype>Normal</papertype>
<papersize>A4</papersize>
<numberofcopies>5</numberofcopies>
<papercolour>Green</papercolour>
<backtoback>false</backtoback>
<collated>false</collated>
<stapled>true</stapled>
<wirobinding>false</wirobinding>
<laminate>false</laminate>
<gluebound>true</gluebound>
<transparencies>false</transparencies>
<twoholepunched>false</twoholepunched>
<folded>false</folded>
<proofrequired>true</proofrequired>
<completed>false</completed>
</printjobform>
<printjobform>
<jobnumber>4</jobnumber>
<userid>2205</userid>
<staffinitials>rwa</staffinitials>
<requestdate>10/04/2010</requestdate>
<copysuppliedas>HARD Copy</copysuppliedas>
<printtype>Black and White</printtype>
<papertype>Card</papertype>
<papersize>A4</papersize>
<numberofcopies>4</numberofcopies>
<papercolour>Purple</papercolour>
<backtoback>false</backtoback>
<collated>true</collated>
<stapled>false</stapled>
<wirobinding>false</wirobinding>
<laminate>false</laminate>
<gluebound>true</gluebound>
<transparencies>false</transparencies>
<twoholepunched>false</twoholepunched>
<folded>false</folded>
<proofrequired>true</proofrequired>
<completed>false</completed>
</printjobform>
</jobs>
```

Comment [C72]: Print job profile

Reprographicssentmessages.xml

```
<?xmlversion="1.0"encoding="utf-8"?>
<reprographicssentmessages>
<message>
<messagenumber>1</messagenumber>
<userid>8000</userid>
<name>2204</name>
<datesent>03/02/2010</datesent>
<topic>test</topic>
<message>This is a test message.</message>
```

```
<messageread>true</messageread>
</message>
<message>
<messagenumber>2</messagenumber>
<userid>2204</userid>
<name>abc</name>
<datesent>03/02/2010</datesent>
<topic>Test</topic>
<message>This is a test email</message>
<messageread>true</messageread>
</message>
<message>
<messagenumber>3</messagenumber>
<userid>2204</userid>
<name>abc</name>
<datesent>03/02/2010</datesent>
<topic>Hello</topic>
<message>Please send a message to me if you have read this message</message>
<messageread>true</messageread>
</message>
</reprographicssentmessages>
```

Comment [C73]: Message profile

Comment [C74]: Depending on the stauts, if 'false' the message has not been read and will be add to the new message stack and if 'true' it will be added to the past message stack as it means the message has already been read.

Specialprices.xml

```
<?xmlversion="1.0"encoding="utf-8"?>
<prices>
<specialprice>
<a3>0.20</a3>
<a4>0.15</a4>
<a5>0.10</a5>
<normal>0.15</normal>
<card>0.20</card>
<blackandwhite>0.20</blackandwhite>
<colour>0.25</colour>
<papercolour>0.35</papercolour>
<paperbandw>0.45</paperbandw>
<backtoback>0.25</backtoback>
<collated>0.30</collated>
<stapled>0.20</stapled>
<wirobinding>0.10</wirobinding>
<laminatea4>0.10</laminatea4>
<laminatea3>0.30</laminatea3>
<gluebound>0.30</gluebound>
<transparencies>0.50</transparencies>
<twoholepunched>0.45</twoholepunched>
<folded>0.40</folded>
</specialprice>
</prices>
```

Comment [C75]: Prices are in pounds (£)

Userdetails.xml

```
<?xmlversion="1.0"encoding="utf-8"?>
<users>
<user>
<userid>8000</userid>
<department>Reprographics</department>
<password>aa</password>
<confirmpassword>aa</confirmpassword>
<securityquestion>What is your name?</securityquestion>
<securityanswer>Reprographics</securityanswer>
</user>
```

Comment [C76]: User profile

```
<user>
<userid>2204</userid>
<department>Art</department>
<password>art</password>
<confirmpassword>art</confirmpassword>
<securityquestion>What is your school name?</securityquestion>
<securityanswer>king edward</securityanswer>
</user>
<user>
<userid>2205</userid>
<department>Biology</department>
<password>hi</password>
<confirmpassword>hi</confirmpassword>
<securityquestion>What is your school name?</securityquestion>
<securityanswer>king edward</securityanswer>
</user>
</users>
```

Section 4

Testing

Test series and numbers	Purpose	Description	Test data	Expected result
1	Security and privacy	Validate to see that the password is hidden when typed and is replaced by '*'	Type some random letters (e.g. abcdefghi)	Letters should be replaced by a '*' symbol.
2	Validate number of copies allowed to be printed	Each department is allowed to print up to and including 700 copies per print job request, anymore should be declined by the system.	699 700 701	Accept value Accept value Decline value (Error message outputted)
3	Validate to see if user ID entered on 'Send message' form is a entered correctly and is checked by the system to confirm	If user enters a User ID that is incorrect, the system will confirm this by checking the list of valid user IDs and if the one entered is not there it should output a error message to the user and also an error message if nothing has been entered	2569 1111 2568 6589 'abs'	Accept Decline (Error message) Accept Decline (Error message) Decline (Error message)
4	Validate output message after print job form has been submitted	Once the user has finished completing the print job form and once they press the 'submit' button, the system should do some last minuet checks and if everything is fine and is sent to the reprographics, a message should be outputted to say that it has been sent. And if the form has not been sent, a message is output displaying reasons to why it has been declined	All field filled correctly Some field corrected correctly Nothing entered on form	Message output ('Your form has been successfully sent') Message output ('Your form has not been sent because the date has not been entered') Message output ('Nothing has been entered)
5	Validate sorting in accounts form	Check to see that data that is wanted is outputted to the reprographics so that if the reprographics wants to view all jobs submitted from Physics department for a certain period of time, the find button does this.	Physics jobs submitted last week Nothing entered Psychology jobs submitted yesterday Chemistry jobs submitted in a future date	All jobs printed by the Physics department is outputted Error Message output ('Nothing has been entered') Error Message output ('Invalid department entered') Error Message output ('Cannot show jobs of future event')

Test 1

The screen shot below confirms that the first test works. The password is hidden as it is typed; this is shown in figure 1.

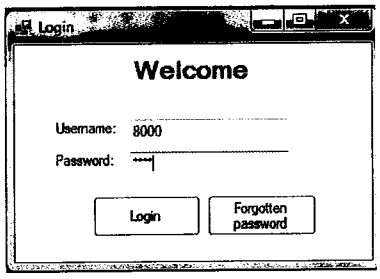
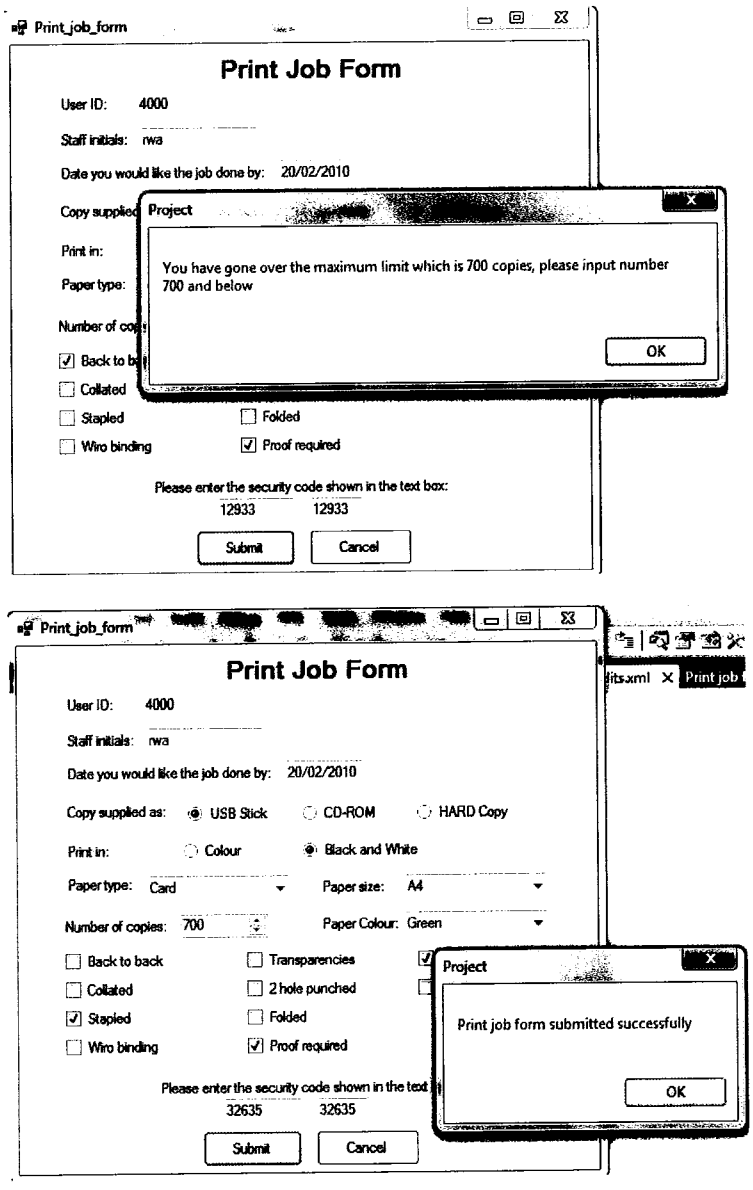
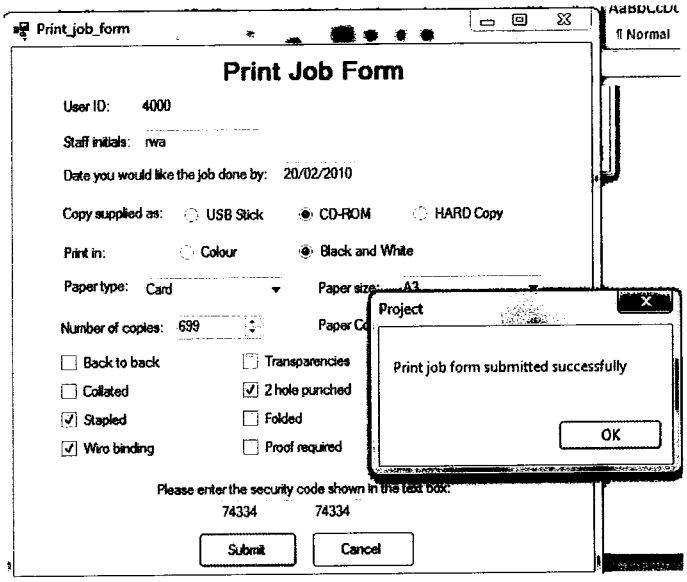


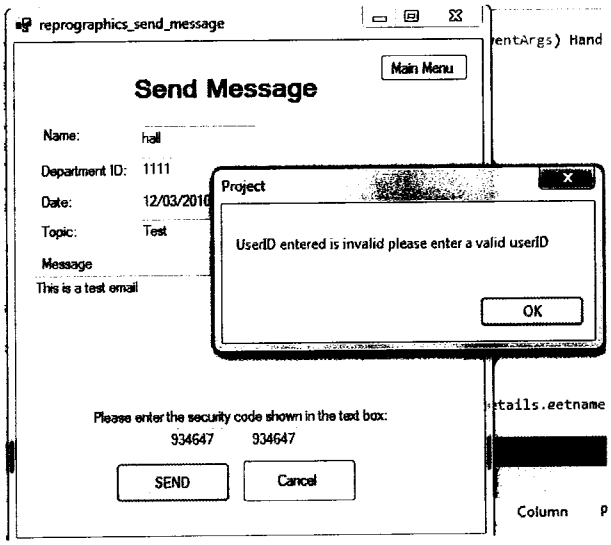
Figure 1

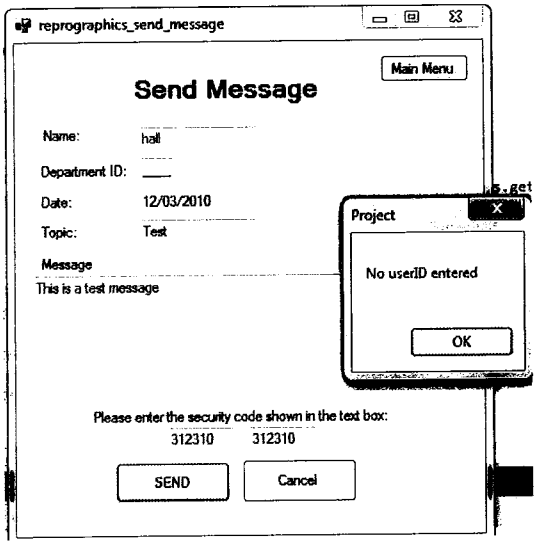
Test 2

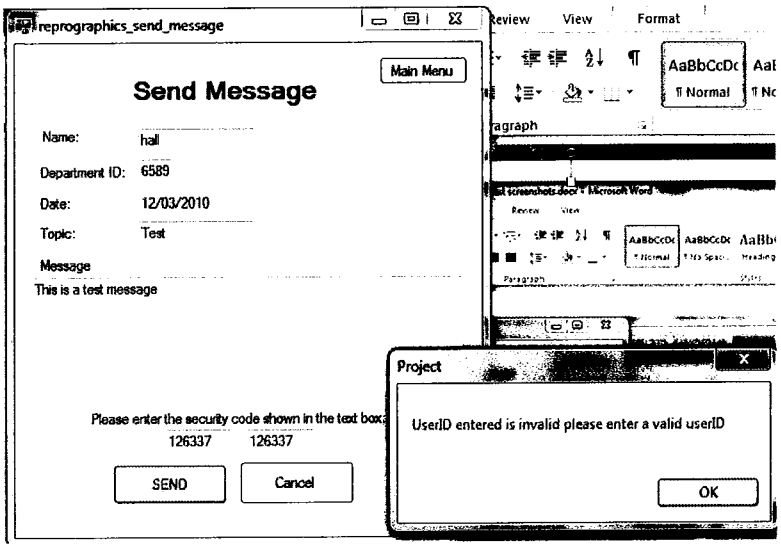


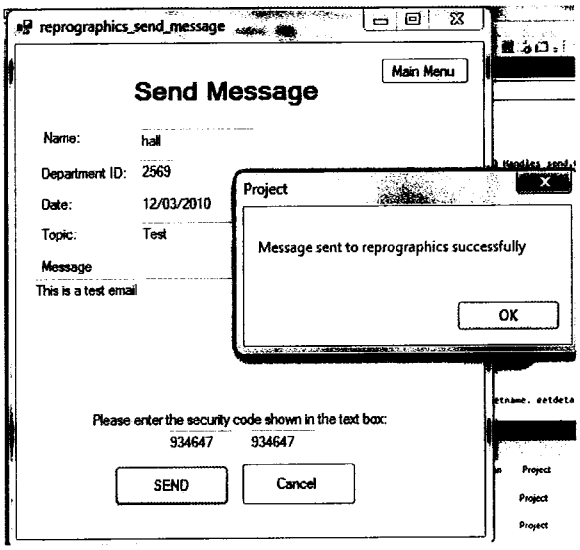

Print_job_form
Print Job Form
User ID: 4000
Staff initials: rwa
Date you would like the job done by: 20/02/2010
Copy supplied as: USB Stick CD-ROM HARD Copy
Print in: Colour Black and White
Paper type: Card
Number of copies: 699
Back to back
Collated
Stapled
Wire binding
Transparencies
2 hole punched
Folded
Proof required
Please enter the security code shown in the text box:
74334 74334
Submit
Cancel
Project
Print job form submitted successfully
OK

Test 3

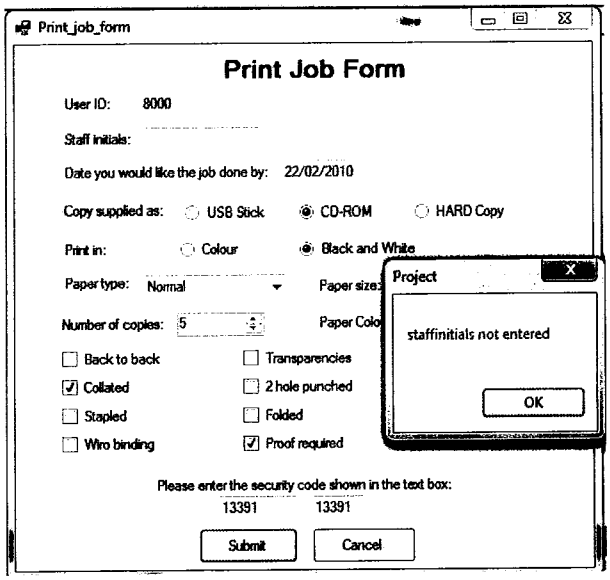

reprographics_send_message
Send Message
Main Menu
Name: hall
Department ID: 1111
Date: 12/03/2010
Topic: Test
Message
This is a test email
Please enter the security code shown in the text box:
934647 934647
SEND
Cancel
Project
UserID entered is invalid please enter a valid userID
OK

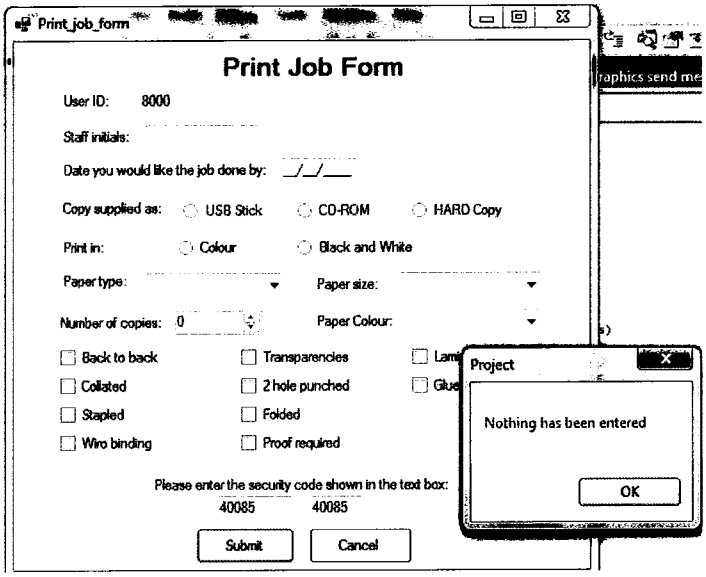

reprographics_send_message
Send Message
Main Menu
Name: hall
Department ID:
Date: 12/03/2010
Topic: Test
Message
This is a test message
Please enter the security code shown in the text box:
312310 312310
SEND
Cancel
Project
No userID entered
OK

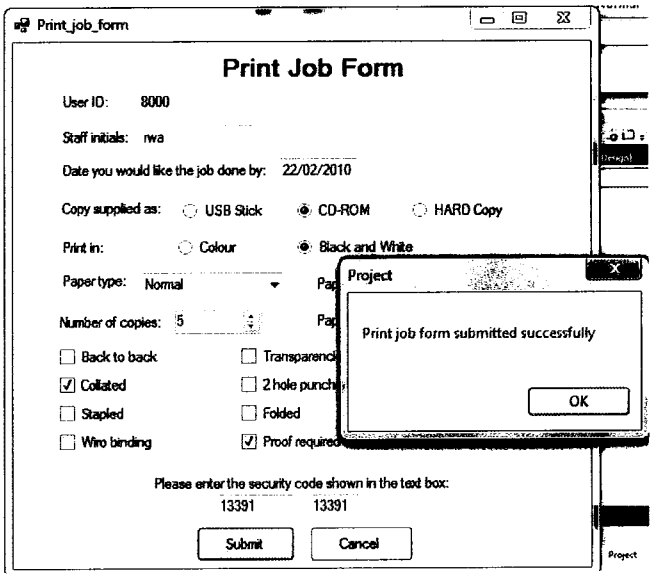

reprographics_send_message
Main Menu
Send Message
Name: hall
Department ID: 6589
Date: 12/03/2010
Topic: Test
Message
This is a test message
Please enter the security code shown in the text box:
126337 126337
SEND
Cancel
Project
UserID entered is invalid please enter a valid userID
OK


reprographics_send_message
Main Menu
Send Message
Name: hall
Department ID: 2569
Date: 12/03/2010
Topic: Test
Message
This is a test email
Project
Message sent to reprographics successfully
OK
Please enter the security code shown in the text box:
934647 934647
SEND
Cancel

Test 4


Print_job_form
Print Job Form
User ID: 8000
Staff initials:
Date you would like the job done by: 22/02/2010
Copy supplied as: USB Stick CD-ROM HARD Copy
Print in: Colour Black and White
Paper type: Normal
Number of copies: 5
Back to back
Collated
Stapled
Wiro binding
Transparencies
2 hole punched
Folded
Proof required
Project
staffinitials not entered
OK
Please enter the security code shown in the text box:
13391 13391
Submit
Cancel


Print_job_form
Print Job Form
User ID: 8000
Staff initials:
Date you would like the job done by: __/__/____
Copy supplied as: USB Stick CD-ROM HARD Copy
Print in: Colour Black and White
Paper type:
Paper size:
Number of copies: 0
Paper Colour:
Back to back
Transparencies
Collated
2 hole punched
Stapled
Folded
Wiro binding
Proof required
Please enter the security code shown in the text box:
40085 40085
Submit
Cancel
Project
Nothing has been entered
OK


Print_job_form
Print Job Form
User ID: 8000
Staff initials: rwa
Date you would like the job done by: 22/02/2010
Copy supplied as: USB Stick CD-ROM HARD Copy
Print in: Colour Black and White
Paper type: Normal
Number of copies: 5
Back to back
Collated
Stapled
Wiro binding
Proof required
Please enter the security code shown in the text box:
13391 13391
Submit
Cancel
Project
Print job form submitted successfully
OK

Section 5

User guide

User Guide

Contents **Page number**

Installation instruction

The system is saved in the saved on the schools shared area and is available on all school computers, the file is located in the start menu under 'Print job', and the file name is called 'King Edward online reprographics'. Click on it and you will see the login screen.

User guide (user)

1. At the beginning you will see the login screen (figure 1) here you enter your username and password that was given to you by the reprographics. Once you have entered your username and password press the login button.

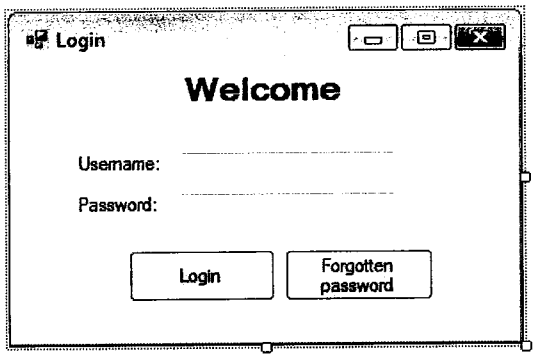


Figure 1

2. After you press the login screen, a message box will appear. If you get the message box similar to figure 2 then go to step 4, if you get the message box that looks like the one in figure 3 then go to step 3 to show you how to retrieve a forgotten password.

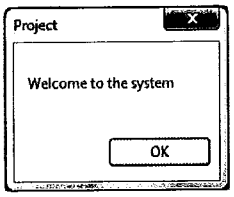


Figure 2

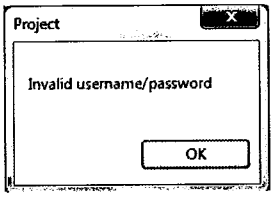


Figure 3

3. On the login screen (figure 1) , click on the button 'Forgotten password' to retrieve your password. After you click on 'forgotten password' you will see the screen shown in figure 4, here you enter the UserID which you will like to retrieve the password for. After doing this click on the 'OK' button.

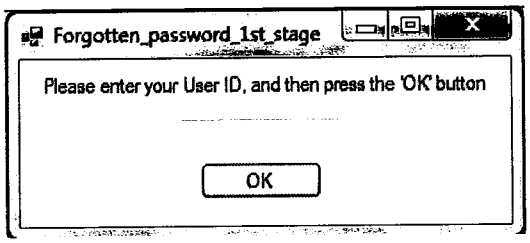


Figure 4

After click on the 'OK' you will get a message box, if the message box says 'Invalid user id' (figure 5) then you will need to see the reprographics teacher. Else if the message box says 'UserID verified' (figure 6) then press the 'OK' button and you will see the next screen similar to figure 7.

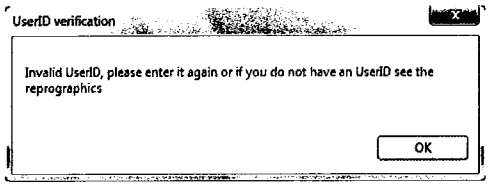


Figure 5

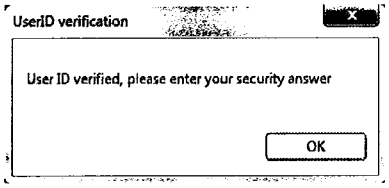


Figure 6

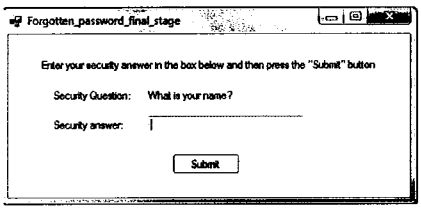


Figure 7

On the 'forgotten password final stage' (figure 7) you put in the answer to the security question you set up with when the reprographics added your UserID. Your security question is already shown above the text box. Once you have entered your password, press the submit button to reveal your password. If you do get an error message saying 'Wrong security answer' please the reprographics teacher for further assistance.

4. Figure 8 below shows the user menu you will get once you have logged it, you have five options to choose, click on any would like to use. The 'useridlabel' will write the name of your department, this will depend on what UserId you logged on with.

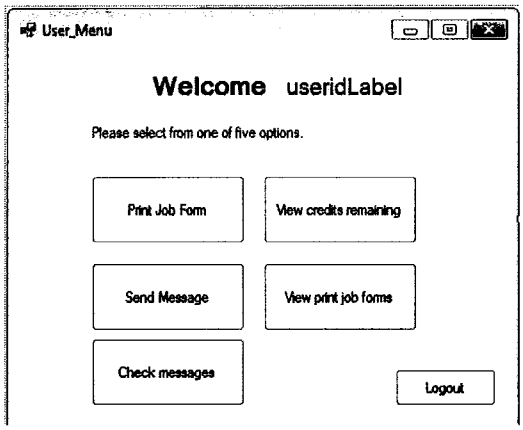


Figure 8

Print job form

Figure 9 shows the print job form. Here you can submit a form which will be sent to the reprographics to be processed.

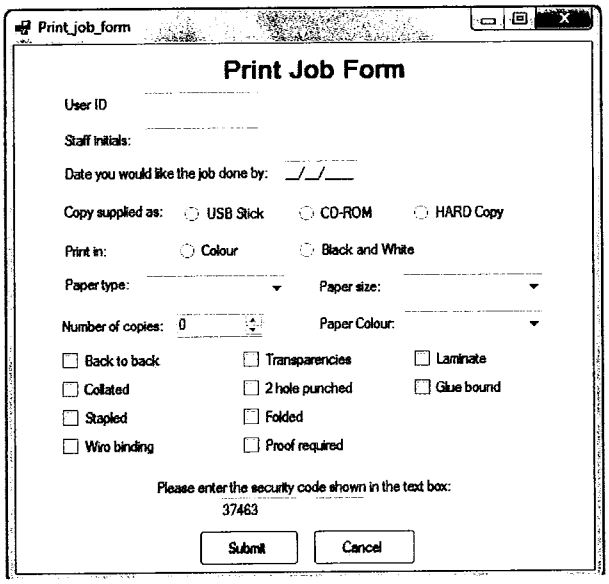


Figure 9

Send Message (figure 10)

Here you can send a message to the reprographics if you have any queries or requests, all you do is just fill in the blanks and then press the 'SEND' button.

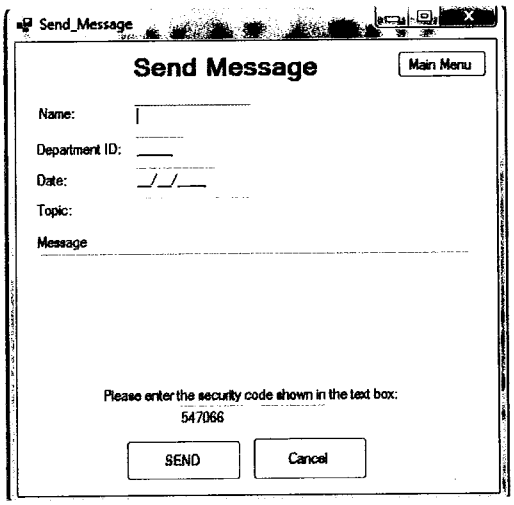


Figure 10

View past print job forms (figure 11)

On this page you can view all the jobs that you or any other member with the same UserID you logged on with has sent in the past. In order to see more information on one of the jobs sent, click on the job you would like more information on and then press 'View selected message' button.

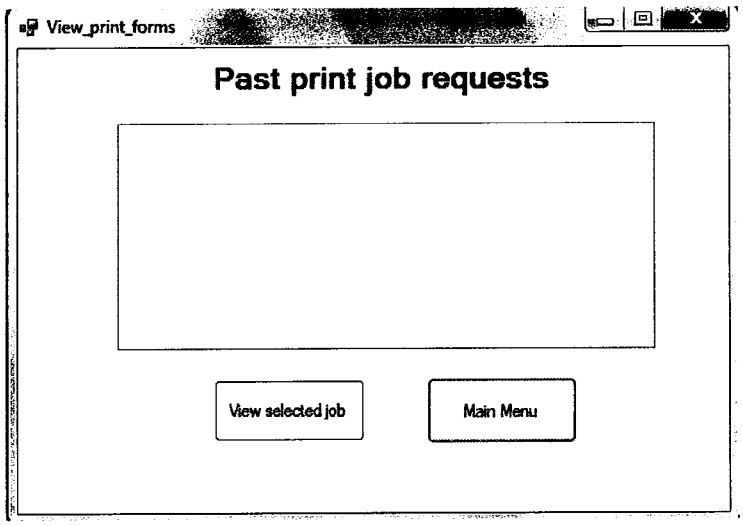


Figure 11

View credits remaining (figure 12)

Here you will be able to see how many credits you have left till the end of the month. If you would like to request for more credits then you have the option to send a message to the reprographics. Or if you would like to see what print job forms you have submitted then you also have the option to go onto that page by clicking on 'view past print job forms submitted' button.

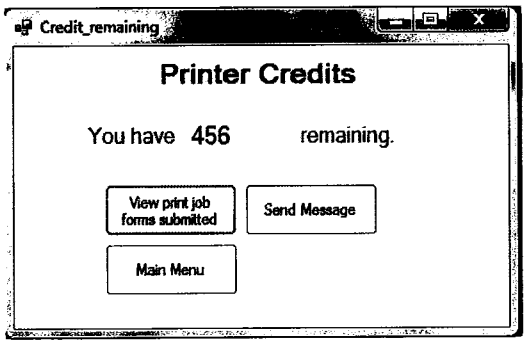


Figure 12

Reprographics user guide

User Guide

1. At the beginning you will see the login screen (figure 13) here you enter your username and password that was given to you by the reprographics. Once you have entered your username and password press the login button.

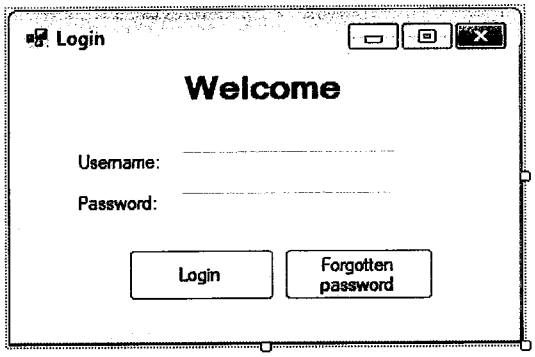


Figure 13

2. After you press the login screen, a message box will appear. If you get the message box similar to figure 14 then go to step 4, if you get the message box that looks like the one in figure 15 then go to step 3 to show you how to retrieve a forgotten password.

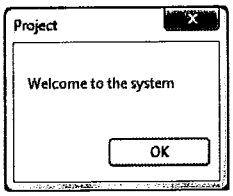


Figure 14

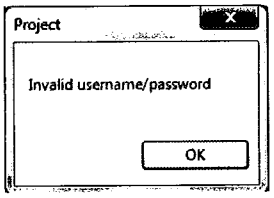


Figure 15

3. On the login screen (figure 13) , click on the button 'Forgotten password' to retrieve your password. After you click on 'forgotten password' you will see the screen shown in figure 16, here you enter the UserID which you will like to retrieve the password for. After doing this click on the 'OK' button.

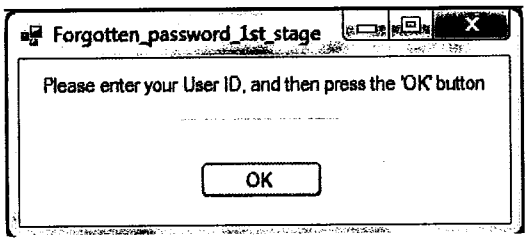


Figure 16

After click on the 'OK' you will get a message box, if the message box says 'Invalid user id' (figure 17) then you will need to see the reprographics teacher. Else if the message box says 'UserID verified' (figure 18) then press the 'OK' button and you will see the next screen similar to figure 19.

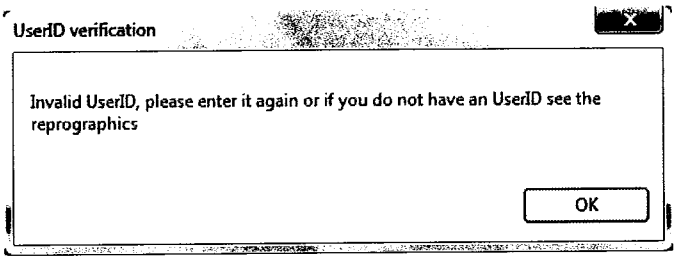


Figure 17

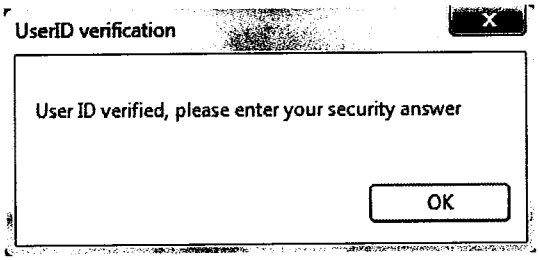


Figure 18

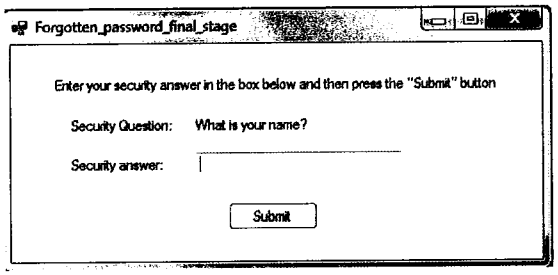


Figure 19

On the 'forgotten password final stage' (figure 19) you put in the answer to the security question you set up with when the reprographics added your UserID. Your security question is already shown above the text box. Once you have entered your password, press the submit button to reveal your password. If you do get an error message saying 'Wrong security answer' please the reprographics teacher for further assistance.

4. Figure 20 on the next page shows the user menu you will get once you have logged it, you have five options to choose, click on any would like to use. The 'useridlabel' will write the name of your department, this will depend on what UserId you logged on with.

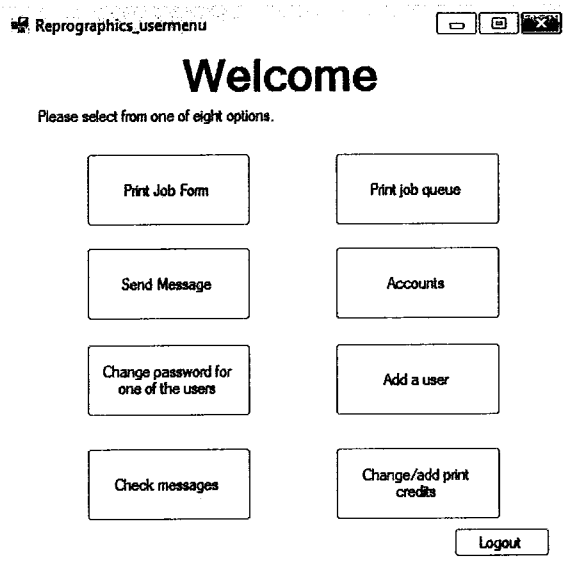


Figure 20

Print job form

Figure 21 shows the print job form where you will be able to submit a job to be processed. All you have to do is fill in all the blanks, the check box options towards the button of the page are optional. If you want to exit the Print Job Form then just press the 'Cancel' button and you will be diverted to the main menu.

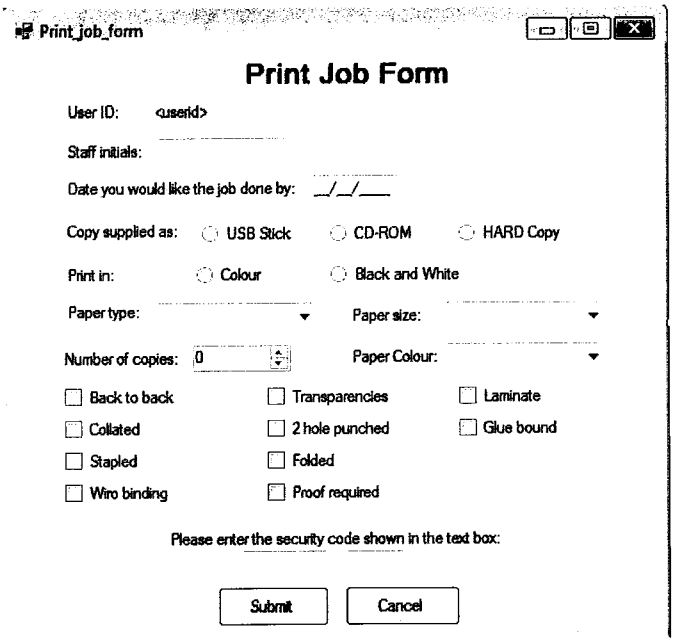


Figure 21

Print job queue (figure 22)

On this 'print job queue' page you can see view all the jobs that have been submitted by everyone. It is a priotised queue sorted by the date they want the job done by. To view a job select the job you want and click on 'View selected job', If you have finished doing the job at the top of the list then click on the 'Remove top job'.

By clicking on the 'update' you can refresh the list to see if there are any new jobs added.

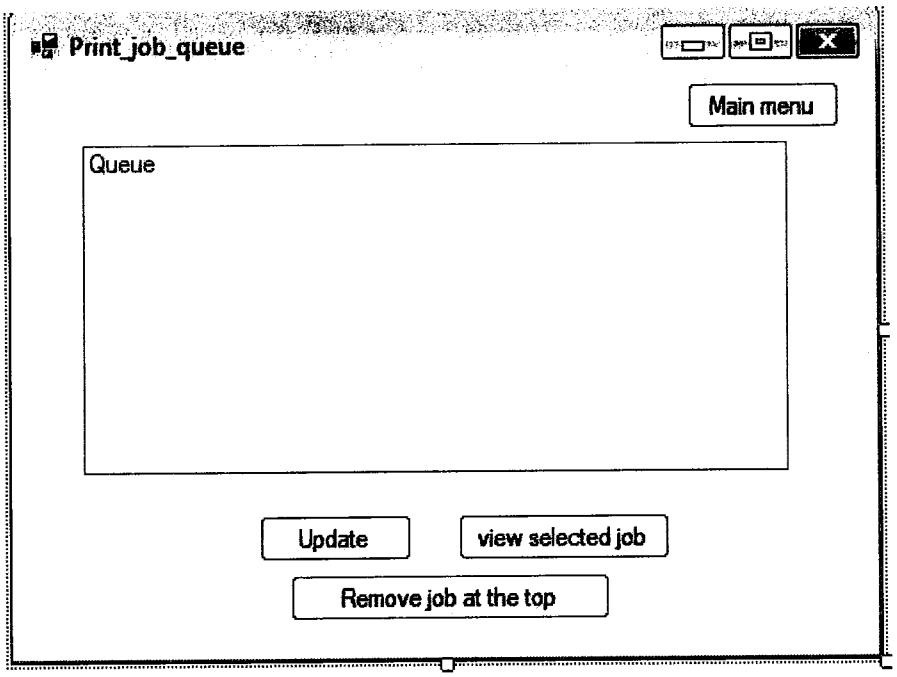


Figure 22

Send Message (figure 23)

This page will allow to send a message to any of the users, all you have to do is fill in the required areas and click on the 'Send' button, this will validate what you have done and if there are any problems the error message will tell you.

To come out of this page just press the 'main menu' button where you will be diverted back to the reprographics menu.

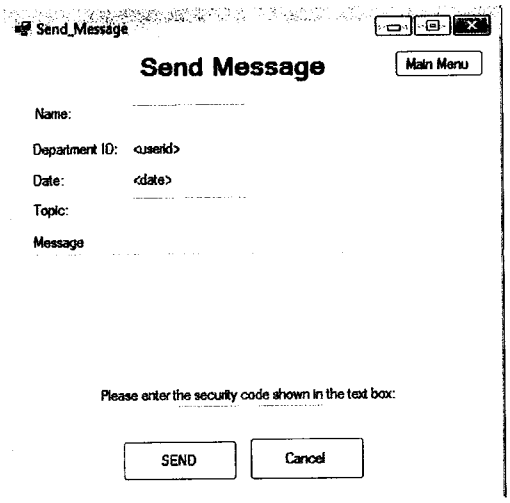


Figure 23

Accounts

This is where you can view all the financial information regarding the job costs. When you click on the 'Accounts' button on the main menu you will be directed to the 'accounts' page that will look like the one shown in figure 24. On the 'accounts' page it shows you all the jobs that have been done with the total cost of each job. In order to view a job just select the one you want and click on the 'view selected job' button.

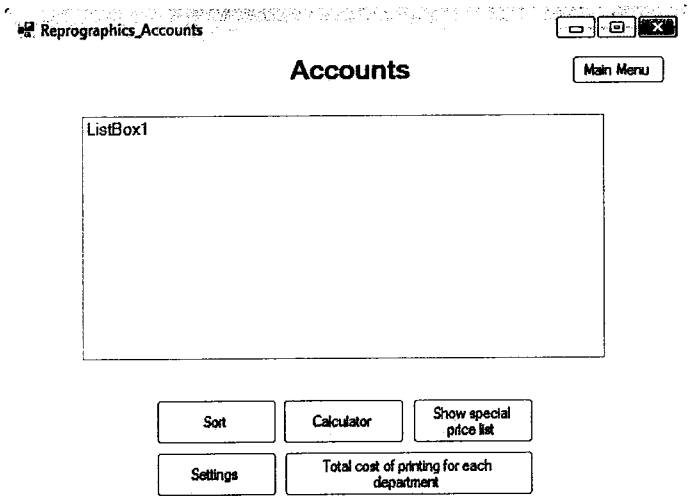


Figure 24

In additon you can see you also have another five options to choose from:

1. Sort
2. Calculator
3. Show special price list
4. Settings
5. Total cost of printing for each department

Sort (figure 25)

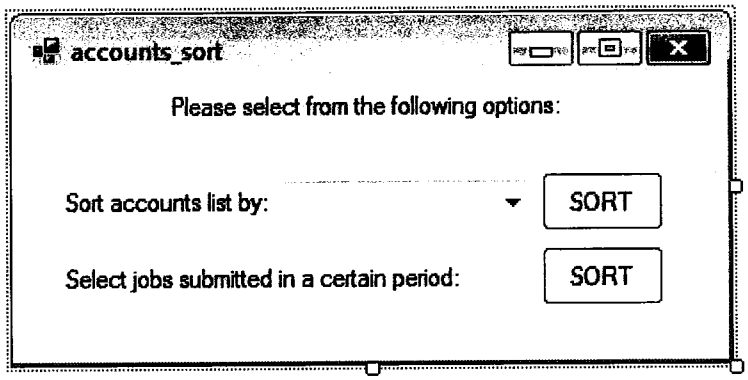


Figure 25

You will be given two options, the first one you select from a drop down list and then click on the 'sort' button, this sorts the whole list which you saw on the 'accounts page'. You can sort the list by:

- ✓ UserID
- ✓ Date job sent
- ✓ Number of copies
- ✓ Total cost

The second option, below the first one allows you to sort the data on the accounts page in a certain period. All you do is click on 'Sort certain period' button and you will then be directed to the 'accounts sort certain period' screen (figure 26). Here you fill in all the dates you want to view the data from and also how you would like to sort the list by.

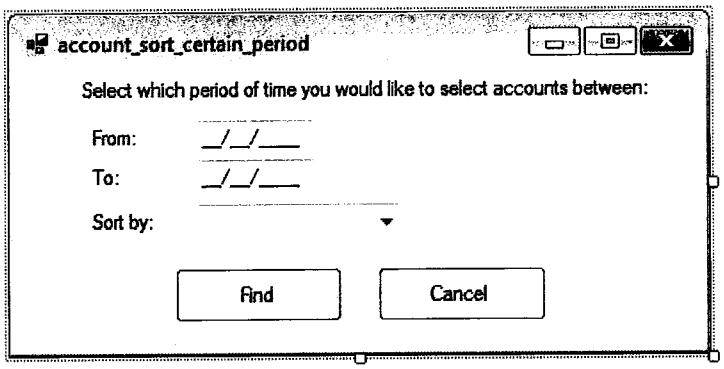


Figure 26

Once you have done that click on the 'sort' button and you will be directed to a page similar to figure 27.

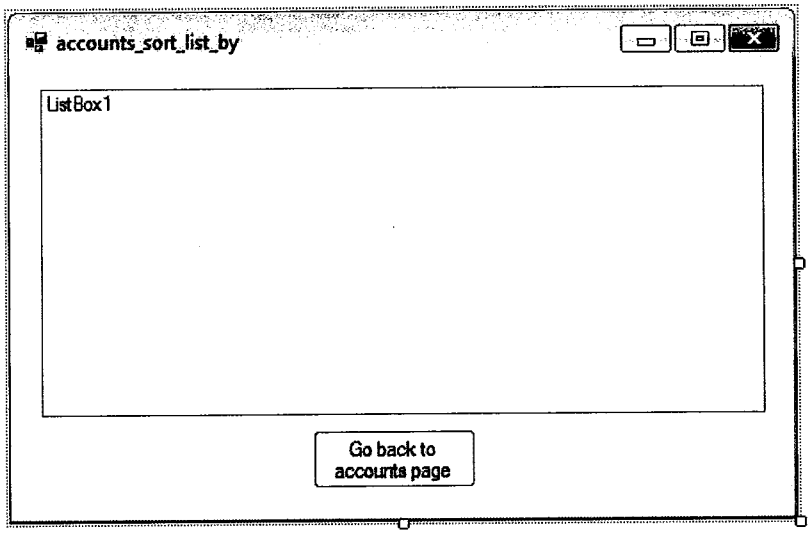


Figure 27

Calculator (figure 28)

This device allows you to make quick simple calculations like addition, subtraction, division and multiplication.

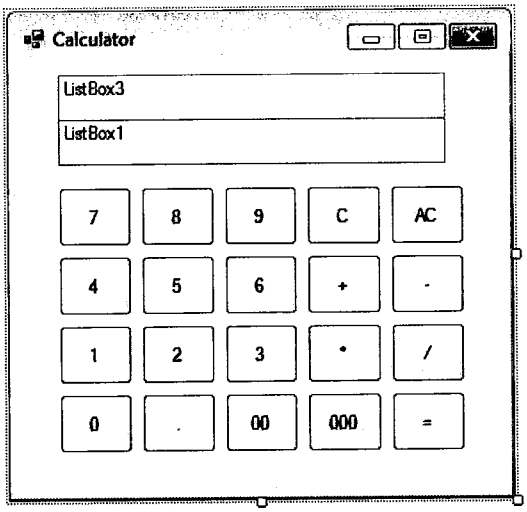


Figure 28

Settings

If you want to make some changes to the prices of special items then this page will allow you to do that. When you click on the 'setting' button on the 'accounts page' you will be directed to the 'accounts settings' page similar to figure 29. Here you will be given two options:

1. Set new prices on everything

2. Change the price of one of the items

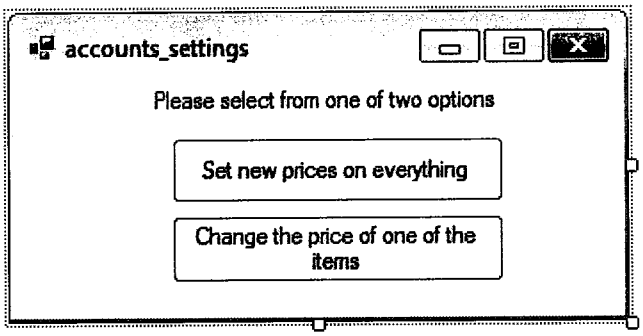


Figure 29

If you want to make changes to everything then click on the 'set neew prices on everything' button where you will be directed to the 'set new prices on everythinhg' page (figure 30). From here you just input the price of everything in pence and then click on the 'set prices' button.

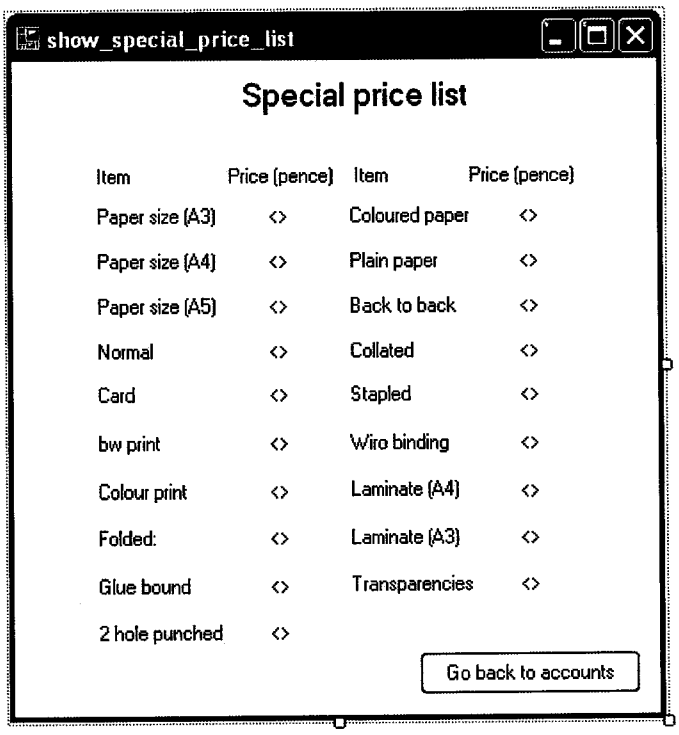


Figure 30

On the other hand if you want to make a change to only one of th special price item, then on the 'accounts setting' page (figure 29) click on the 'change the price of one of the items' button where you will then be directed to the 'change one special pricew item' page (figure 31). Here select the special price item from the drop down box and then input the new price in pence in the box below. Then click on 'change price' where a message will be outputted saying 'saying price changed'.

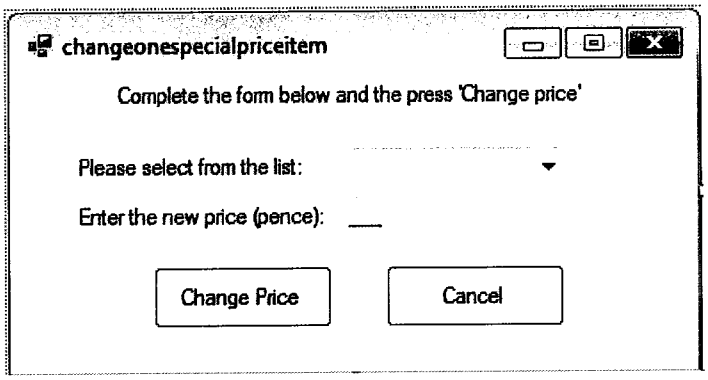


Figure 31

Show special prices (figure 32)

This page allows you to view all the prices of the special prices.

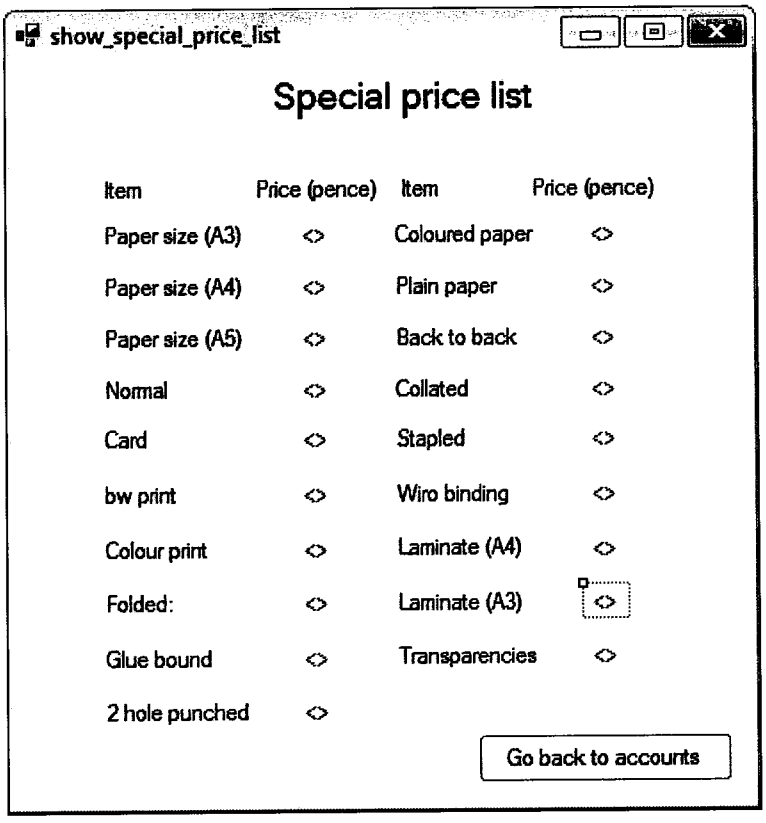


Figure 32

Total cost of each department (figure 33)

This page will allow you to view the toal cost of each department so far.

Change password for one of the users (figure 34)

This page will allow you to change one of the users passwords, all you have to do is input the user and their current password, fllowed by their new passoword and after that click on the 'change password' button.

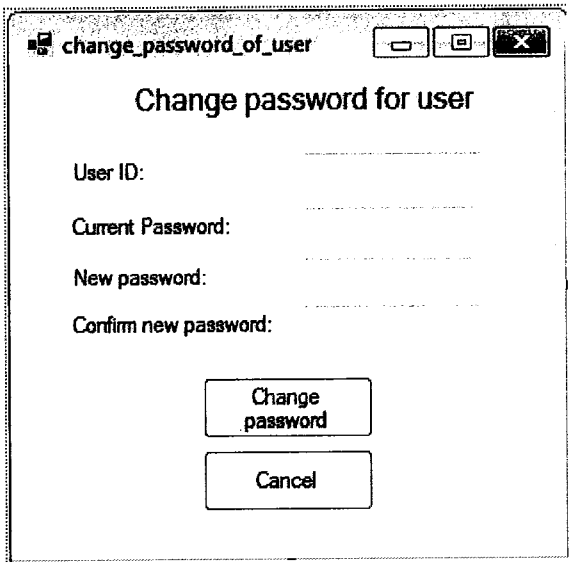


Figure 34

Add a user (figure 35)

This page will allow you to creat or add a new user to the system. All you have to do is fill in the details in each box. The UserID is validated so that only numbers can be inputted. After you have filled in all the details click on 'create user' and you should then receive a message box saying 'User created', if there is a problem then the message box will give clear reasons to why the new user has not been created.

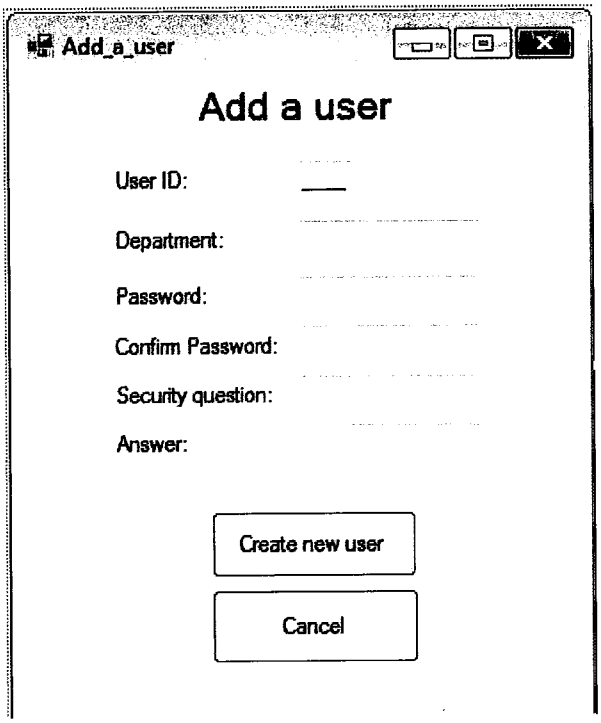


Figure 35

Check messages

If you have got any messages then you will be able to view both your new messages and also you past messages. The first screen you get when you click on 'Check messages' is new message screen (figure 36). If you have any new messages you will be able to view them simply by selecting the message and then clicking on the 'view selected message' button. You should then get a screen similar to figure 37.

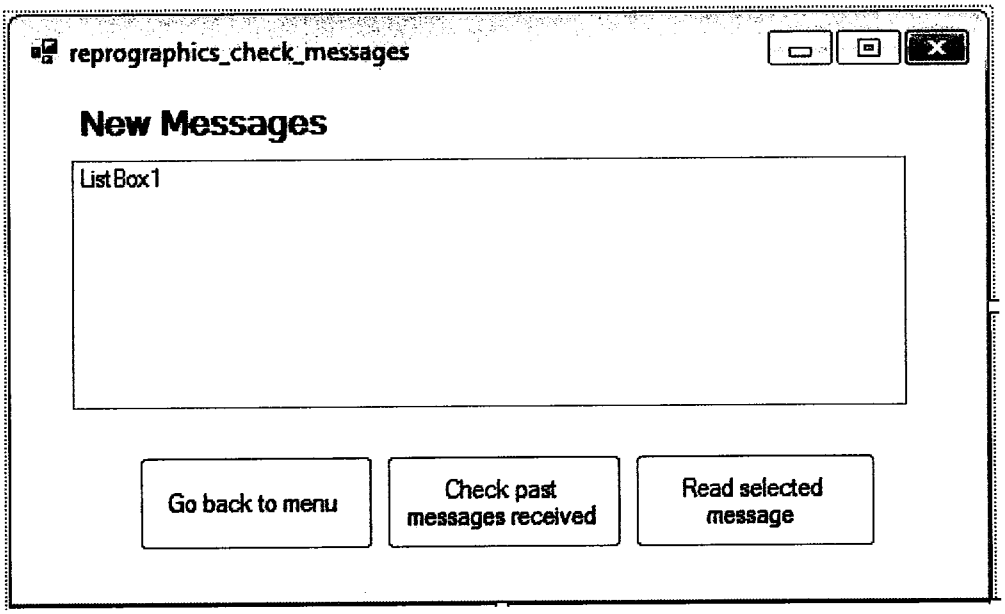


Figure 36

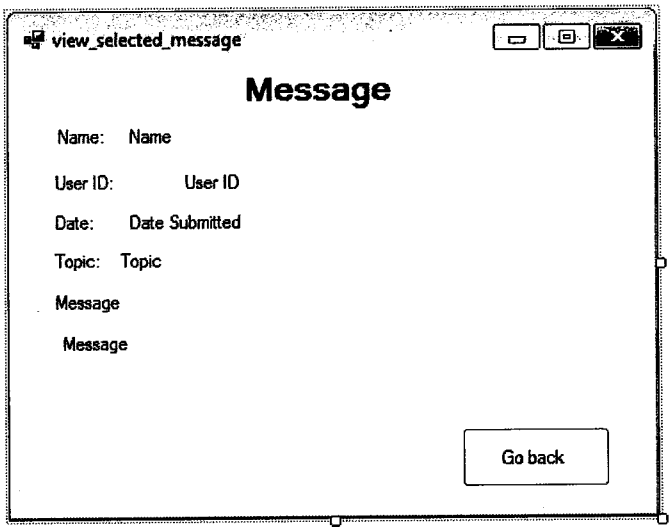


Figure 37

If you have got no messages but you want to look at your past messages then you can do so by clicking on 'view past messages' button which is on the new message screen. YOu will then see a screen similar to figure 38.

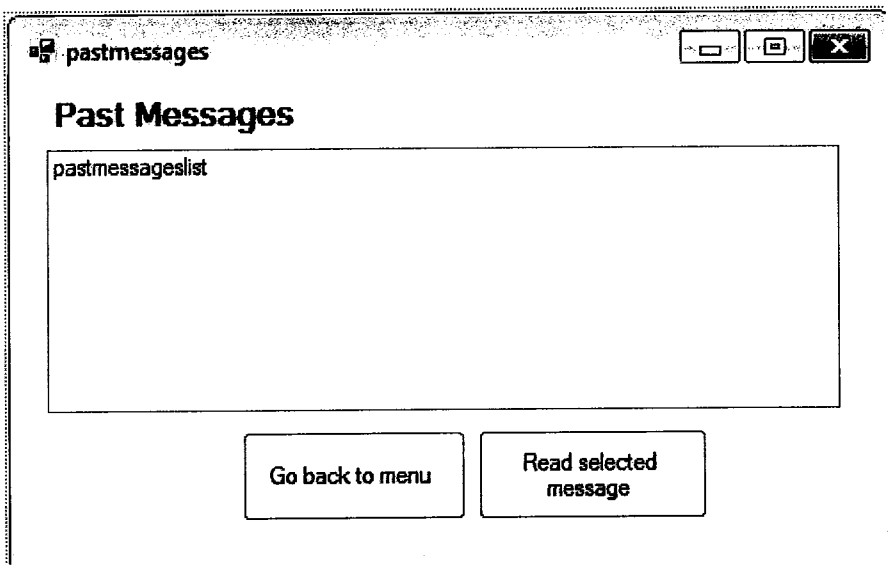


Figure 38

Change/add print credits

To add/change print credits:

1. Click on the 'check/add print credits' button on the main menu
2. You will see a screen like the one shown in figure 39

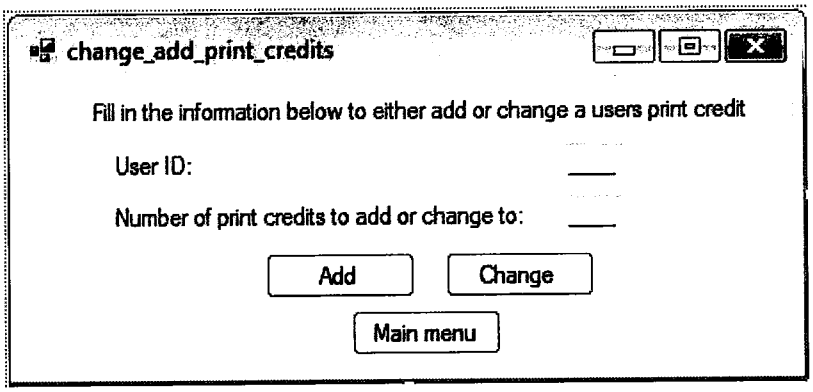


Figure 39

3. Input the userID you want to work with
4. Input the number of credits to add or change to

If you want to add credits to the prievious amount on the users account then click on 'add', or if you wan to change the number of credits they have to a new amount then click 'change'

.

Section 6

System maintenance

System overview

Every thing has been done in small parts consisting of functions, procedure, public and private classes. The benefit of this is so that when you come to correct any section of the system it is easy to track where you are and what needs sorting out.

All objectives which were stated in the analysis section have been me and these have been cross reference in the Implementation section.

All screens have been designed so that it can be easily read. The choice of background colours, font, colour of text have all been carefully designed so that the screens are easy to read. The design of eac page has been done so that it is easy to understand what is happening on each screen and is easy to navigate round the whole system.

Below shows a table of the classes used and the purpose of each one.

Class: Print job form

Function/procedure name	Purpose of function/procedure	Variables used in function/procedure	Purpose of variables
Job complete	Get the job which is at the top of the print job queue and changes its status on the XML file as job completed	Job number (integer)	Opens XML file 'printjobform.xml' Finds the job node number on the XML file that needs to be changes using the class 'find job' Stores the job number node that was found from the find job class
add	Saves the print job form details to the XML file 'print jobform.xml'	Print job form xml	Opens XML file 'printjobform.xml'
length	See how many print job forms have been submitted	Printjob Printjobformitems	Opens XML file a list of printjobforms which are a child of the job node

The following functions each get the information (stated by their name) from the XML file depending on which child node number is required. So for example, child number 2 would select the third print job form (as XML files start from 0) and from that selects what the information that would be required (e.g. staff initial).

Job number
User ID
Staff initials
Request date
Copy supplied as
Paper type
Paper size
Number of copies
Paper colour
Back to back
Collated
Stapled
Wirobinding
Laminate
Glue bound
Transparencies
Two hole punched
Folded

Class: Special price list

Function/procedure name	Purpose of function/procedure	Variables used in function/procedure	Purpose of variables
Change special price list	Gets the selected item that the reprographics to the change the price to and updates the price using what the reprographics used when they fill in the form	Special prices	Opens XML file 'special prices.xml'
add	Saves the special price details to the XML file 'specialprices.xml'	Special price xml	Opens XML file 'specialprices.xml'
Calculate total price of print job	This function calculates the total cost of the job using what the user has just submitted from the print job form.	Totalcost (string)	Stores the total cost

The following functions each get the information (stated by their name) from the XML file depending on which child node number is required:

Papersizea3
Papersizea4
Papersizea5
Normal
Card
Paper colour
Bwprint
Colour print
Plain paper
Backtoback
Collated
Stapled
Wirobinding
Laminatea4
Laminatea3
Gluebound
Transparencies
Twoholepunched
folded

The following procedures change the price of the special item and replace the initial price in the XML file 'Specialprices.xml':

A3
A4
A5
Normal
Card
Balackandwhite
Colour
Paper colour
Paper bandw
Backtoback
Collated
Stapled
Wirobinding
Laminatea4
Laminatea3
Gluebound
Transparencies
Twoholepunched
folded

Class: Print credits

Function/procedure name	Purpose of function/procedure	Variables used in function/procedure	Purpose of variables
Saveprintcredits	To collect the required details from the print job form and add and save it to the Print job credits XML file	n/a	n/a
Update print credits	Opens the XML file 'printcredits.xml' and updates the print credits for that user that has just submitted a form. It also check to see if it has a enough credits to proceed with the print job form they have just	Current print credit number	Stores the new value of print credits
Length	See how many print credit profiles there are	Message Print credit details	Opens XML file a list of print credit which are a child of the user print credit profile node

The following functions each get the information (stated by their name) from the XML file depending on which child node number is required:

UserID
Printcredits

Class: user profile

Function/procedure name	Purpose of function/procedure	Variables used in function/procedure	Purpose of variables
Save user	Saves user details to the XML file 'userdetails.xml'	Usersxml	Opens XML file 'userdetails.xml'
length	See how many print credit profiles there are	Userdetails userdetail	Opens XML file a list of user which are a child of the users node

The following functions each get the information (stated by their name) from the XML file depending on which child node number is required:

UserID
Department
Password

Securityquestion
Securityanswer

The 'change password' procedure is used to change the password of the user and replace this with the initial one in the XML file.

Class: User sent message

Function/procedure name	Purpose of function/procedure	Variables used in function/procedure	Purpose of variables
Message read	Set the me message status to read and saves this information in the XML file	Messagexml Message number	Opens XML file 'messages.xml' Used to find the child node number in the XML file 'messages.xml' of the message profile
Add message	Saves the message form details to the XML file 'print messages.xml'	Message details xml	Opens XML file 'messages.xml'
lengths	See how many messages there are in the XML file 'messages.xml'	Message Message details	Opens XML file 'reprographicssentmessages.xml' a list of message which are a child of the messages node

The following functions each get the information (stated by their name) from the XML file depending on which child node number is required:

Message number
UserID
Name
Date submitted
Topic
Submittedmessage
Read

Class: Reprographics sent message

Function/procedure name	Purpose of function/procedure	Variables used in function/procedure	Purpose of variables
Set message to read	Set the me message status to read and saves this information in the XML file	Messagexml Message number	Opens XML file 'messages.xml' Used to find the child node number in the XML file 'messages.xml' of the message profile
Save reprographics sent message	Saves the message form details to the	Reprographics message sent	Opens XML file 'messages.xml'

	XML file 'print messages.xml'	details xml	
lengths	See how many messages there are in the XML file 'messages.xml'	Message Message details	Opens XML file 'reprographicssentmessages.xml' a list of message which are a child of the reprographics sent messages node

The following functions each get the information (stated by their name) from the XML file depending on which child node number is required:

Message number
UserID
Name
Date submitted
Topic
Submittedmessage
Read

Class: accounts

Function/procedure name	Purpose of function/procedure	Variables used in function/procedure	Purpose of variables
Save account details	Saves the accouts details to the XML file 'accountdetails.xml'	User account details xml	Opens XML file 'accountdetails.xml'
Length	See how many messages there are in the XML file 'messages.xml'	Message Message details	Opens XML file 'accountdetails.xml' a list of user account details which are a child of the accounts node

The following functions each get the information (stated by their name) from the XML file depending on which child node number is required:

UserID
Staff initials
Department
Date submitted
Number of copies
Total price

Class: Queue

The 'add form to queue', initialises the print job queue, clears the queue items, adds items to the queue, removes items from the queue, sorts the contents of the queue and outputs the queue contents.

Each of the procedures below, initial the stack, clears the stack, adds item to the stack, removes items from the stack and outputs the stack contents:

- Add account details
- Add new message to stack
- Add past message to queue
- Add to list past jobs
- Add user past message to stack
- Add user new message to stack

The procedures below store the information collected from the 'send message', 'send message (reprographics)', 'login screen', 'print credit', 'print job form' and 'special price items' screens and stores them in variables where they are used for example to save the information to the XML file. The procedures use 'property get':

- Get change price details (Gets details from the 'change price details' form and stores them in the variables so that they can be used by the other class to perform the rest of the procedure)
- Get user details (Gets details from the 'add new user' form and stores them in the variables so that they can be used by the other class to perform the rest of the procedure (uses property get function))
- Get details for print credit change (Gets details from the 'change print credits' form and stores them in the variables so that they can be used by the class 'print credits' to perform the rest of the procedure (uses property get function))
- Get entered login details (Gets details from the 'login screen' form and stores them in the variables so that they can be used by the class 'user profile' to perform the rest of the procedure (uses property get function))
- Get message details (Gets details from the 'send message' form and stores them in the variables so that they can be used by the class 'user sent message' to perform the rest of the procedure (uses property get function))
- Get new prices on everything (Gets details from the 'change all special item prices' form and stores them in the variables so that they can be used by the class 'special price list' to perform the rest of the procedure (uses property get function))
- Get security answer (Gets details from the 'forgotten password final stage' form and stores them in the variables so that they can be used by the class 'user profile' to perform the rest of the procedure (uses property get function))
- Get print job form items (Gets details from the 'print job form' and stores them in the variables so that they can be used by the other class 'print job form' to perform the rest of the procedure (uses property get function))

The function 'does user have a new message' is uses the class 'user sent messages' and ' reprographics sent messages' to see if there are any messages in the XML file that has not been read.

The 'find user' and 'find job' functions uses the class 'userprofile' and 'print job form' to extract the required information and see if for example the login details mathch what is stored in the XML files or if for example to find the job profile when the reprographics wants to view the details of the print form job.

Section 7

Evaluation

The feedback received from the end user was positive saying that the objectives were met and that the end product was what she was wanted. The end product was implemented in the time period set.

The whole system was tested thoughrely in order to make sure there were no problems. Black-box testing, top down, bottom up and white-box testing were all made. Normal erroneous and boundary were also made. These were all shown in section 4.

The user guide was shown to some of the users to see if the system was not complex for them. Positive feedback was received saying that the system was very easy to use and they could see a big difference to how the new system was different to the original one. Feedback was received saying that the new system was much more efficient and less time consuming.

Communication at all times was made with the end user in order to make sure the end product that was being designed was what she wanted. Some improvements were made after receiving feedback from the end user.

For example, the 'send message' form at the beginning looked like in figure 1.

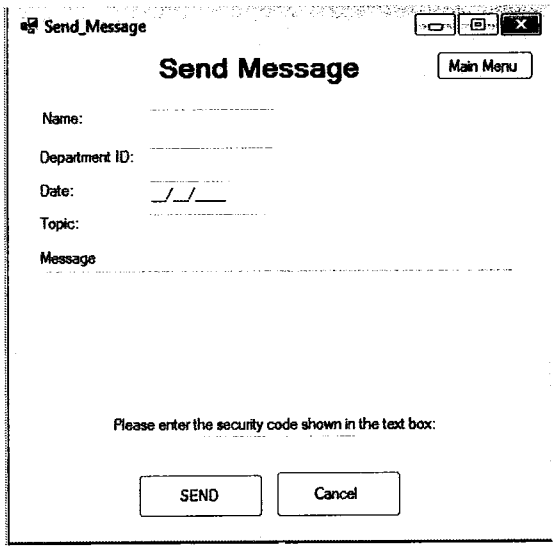


Figure 1

This was later changed to figure 2. Improvements made were so that when the user opens the 'send message' form, the date and userID are already filled in, this decreased the time taken to send a message making the whole process of sending a message more efficient. Also so the user or someone who had managed to gain access did not mess up the system by keeping sending messages to the reprographics a security code was added so that each time they sent a message they had to enter a security code, this stopped the likely hood of spam.

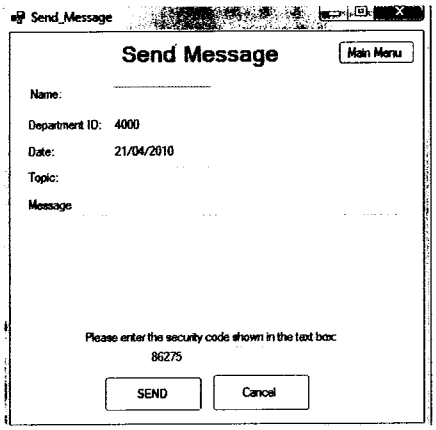


Figure 2

The same same changes were made to the print job form as figure 3 and figure four shows:

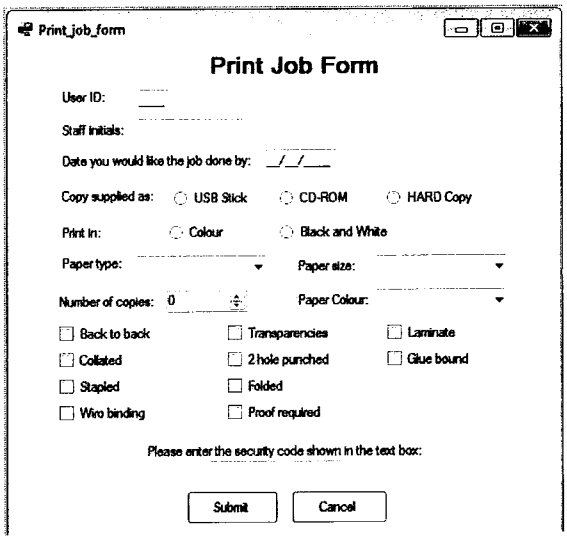


figure 3

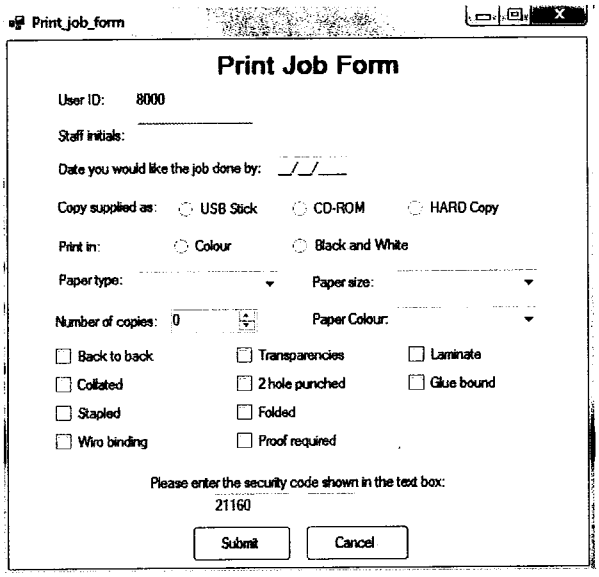


Figure 4

Improvements that can be done to the system

One improvement that can be done is to give the option to the user itself to change their password.

Another improvement is to code the password so that if someone did gain access to the XML file, they won't be able to see what the password is. And whenever the system wants to match for example the entered password in the login screen with the password in the XML file, the password in the SML file will first be decoded and then checked.

Sources

This websites and books were used along side the implementating stage in order to see how to do certain things, like for example how to design windows forms:

- Learn Visual basic in 24 hours
- MSDN website
- Microsoft website

Established 1553

[redacted]

School

[redacted]

Dear [redacted],

Re: Project from September 2009 - March 2010

I have been impressed with the amount of thought that has gone into developing the print management system and this has meant that the product serves the needs of the operator very well. It is particularly difficult to create a system that is user friendly, without a thorough understanding of how it is to be used in practice and therefore some moderations have been made in response to operator input. The final version is one that should prove simple to use, as well as more effective in producing account invoices.

Yours faithfully

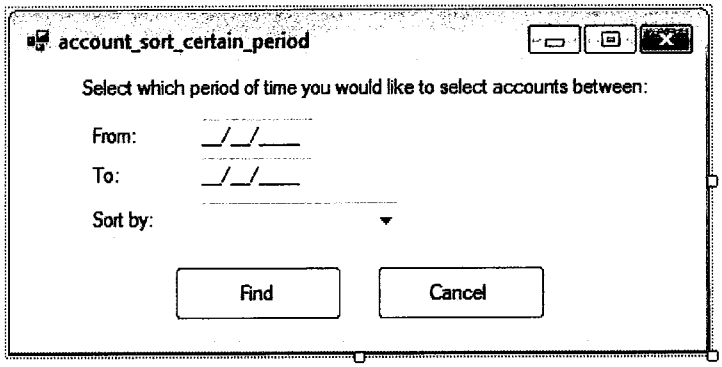
[redacted]

[redacted]

Reprographics Technician

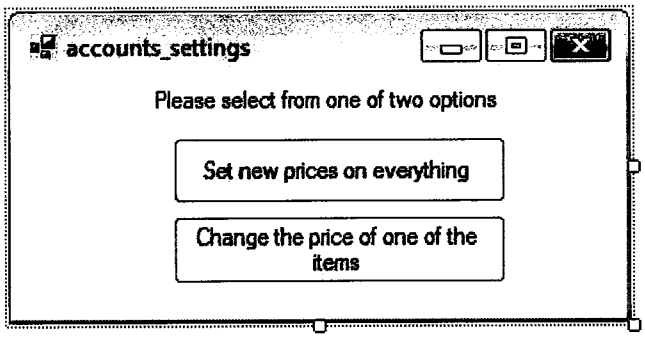
Table of Contents

Account sort certain period



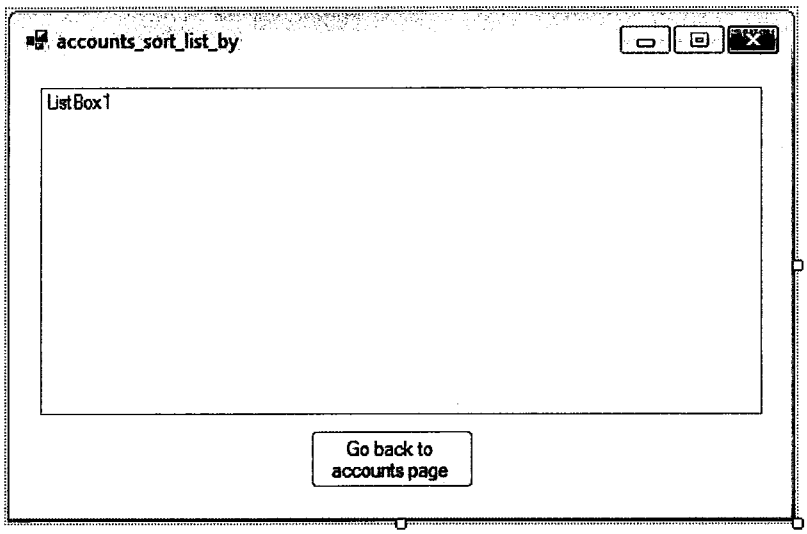

```
Public Class account_sort_certain_period

    Private Sub cancel_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cancel.Click
        Reprographics_Accounts.Show()
        Me.Close()
    End Sub

    Private Sub find_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles find.Click
        show_accounts_certain_period.Show()
        Me.Visible = False
    End Sub

End Class
```

Accounts settings




```
Public Class accounts_settings

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        changeonespecialpriceitem.Show()
        Me.Close()
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Set_new_prices_on_everything.Show()
        Me.Close()
    End Sub
End Class
```

Accounts sort list by

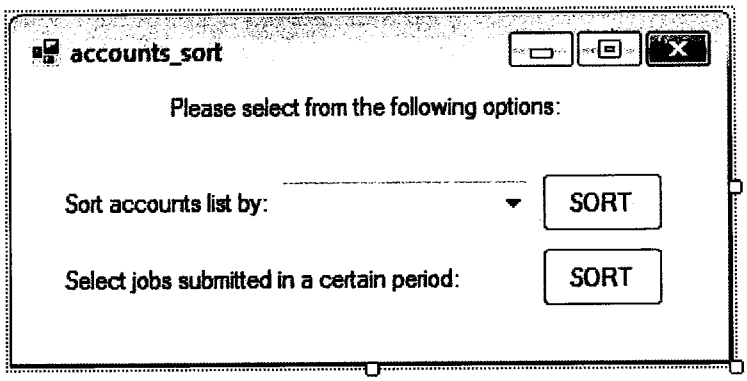



```
Public Class accounts_sort_list_by

    Private Sub accounts_sort_list_by_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim length As Integer
        Dim number As Integer = 0
        Dim accounts As New accounts
        Dim type As String
        type = accounts_sort.sortoptions.Text
        length = accounts.Length
        Dim array(length) As Integer
        Dim value As Integer = 1
        Dim temporystoredvalue As Integer
        Dim temporystoreddate As String
        Dim a As String
        Dim nomoreswaps As Boolean = True
        Dim usernumber As Integer
        Dim find As New findjob
        Dim datesubmitted(length) As String
        If type = "User ID" Then
            For value = 1 To length
                array(value) = accounts.userid(number)
                a = array(value)
                number = number + 1
            Next
            Do

                nomoreswaps = True
                For value = 1 To length - 1
                    If array(value) > array(value + 1) Then
                        nomoreswaps = False
                        temporystoredvalue = array(value)
                        array(value) = array(value + 1)
                        array(value + 1) = temporystoredvalue
                    End If
                Next
            Loop Until nomoreswaps = True
            For value = 1 To length
                usernumber = find.findrightaccountuseriddetails(array(value))
```

```
                ListBox1.Items.Add(accounts.userid(usernumber) & "        " &
accounts.staffinitials(usernumber) & "                " &
accounts.numberofcopies(usernumber) & "                " &
accounts.datesubmitted(usernumber) & "            £" &
accounts.totalprice(usernumber))
            Next
        ElseIf type = "Number of copies" Then
            For value = 1 To length
                array(value) = accounts.numberofcopies(number)
                a = array(value)
                number = number + 1
            Next
            Do

                nomoreswaps = True
                For value = 1 To length - 1
                    If array(value) > array(value + 1) Then
                        nomoreswaps = False
                        temporystoredvalue = array(value)
                        array(value) = array(value + 1)
                        array(value + 1) = temporystoredvalue
                    End If
                Next
            Loop Until nomoreswaps = True
            For value = 1 To length
                usernumber = find.findrightaccountnumberofcopiesdetails(array(value))
                ListBox1.Items.Add(accounts.userid(usernumber) & "        " &
accounts.staffinitials(usernumber) & "                " &
accounts.numberofcopies(usernumber) & "                " &
accounts.datesubmitted(usernumber) & "            £" &
accounts.totalprice(usernumber))
            Next
        ElseIf type = "Total cost" Then
            For value = 1 To length
                array(value) = accounts.totalprice(number)
                a = array(value)
                number = number + 1
            Next
            Do

                nomoreswaps = True
                For value = 1 To length - 1
                    If array(value) > array(value + 1) Then
                        nomoreswaps = False
                        temporystoredvalue = array(value)
                        array(value) = array(value + 1)
                        array(value + 1) = temporystoredvalue
                    End If
                Next
            Loop Until nomoreswaps = True
            For value = 1 To length
                usernumber = find.findrightaccounttotalcostdetails(array(value))
                ListBox1.Items.Add(accounts.userid(usernumber) & "        " &
accounts.staffinitials(usernumber) & "                " &
accounts.numberofcopies(usernumber) & "                " &
accounts.datesubmitted(usernumber) & "            £" &
accounts.totalprice(usernumber))
            Next
        ElseIf type = "Date submitted" Then
            For value = 1 To length
                datesubmitted(value) = accounts.datesubmitted(number)
                number = number + 1
            Next
            Do
```

```
                nomoreswaps = True
                For value = 1 To length - 1
                    If SortableDate(datesubmitted(value)) >
SortableDate(datesubmitted(value + 1)) Then
                        temporystoreddate = datesubmitted(value)
                        datesubmitted(value) = datesubmitted(value + 1)
                        datesubmitted(value + 1) = temporystoreddate
                        nomoreswaps = False
                    End If
                Next
            Loop Until nomoreswaps = True
            For value = 1 To length

                usernumber =
find.findrightaccountdatesubmitteddetails(datesubmitted(value))
                ListBox1.Items.Add(accounts.userid(usernumber) & "        " &
accounts.staffinitials(usernumber) & "                " &
accounts.numberofcopies(usernumber) & "                " &
accounts.datesubmitted(usernumber) & "            £" &
accounts.totalprice(usernumber))
            Next
        End If
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Reprographics_Accounts.Show()
        Me.Close()
    End Sub
    Function SortableDate(ByVal Datevalue As String)
        Dim reorderdate As String
        reorderdate = Datevalue.Substring(8, 2) & Datevalue.Substring(3, 2) &
Datevalue.Substring(0, 2)
        SortableDate = reorderdate
        Return SortableDate
    End Function

End Class
```

Accounts sort




```
Public Class accounts_sort

    Private Sub sortlist_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles sortlist.Click
        accounts_sort_list_by.Show()
```

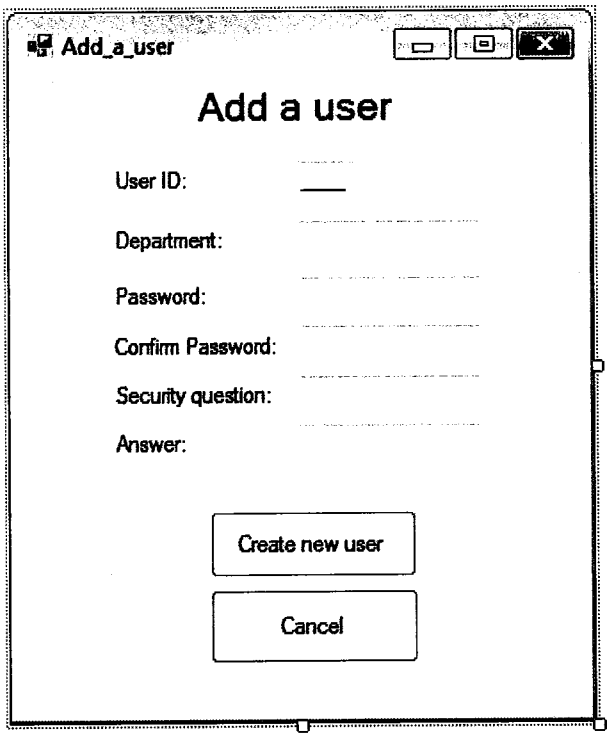
```
        Me.Visible = False
    End Sub

    Private Sub sortcertainperiod_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles sortcertainperiod.Click
        account_sort_certain_period.Show()
        Me.Close()
    End Sub

End Class
```

Accounts (class)

```
Public Class accounts
    Function userid(ByVal number As Integer)
        Dim accounts As New System.Xml.XmlDocument
        accounts.Load("accountdetails.xml")
        Dim accountdetails As System.Xml.XmlNodeList =
accounts.SelectNodes("accounts/useraccountdetails")
        userid = accountdetails.Item(number).SelectSingleNode("userid").InnerText()
    End Function
    Function staffinitials(ByVal number As Integer)
        Dim accounts As New System.Xml.XmlDocument
        accounts.Load("accountdetails.xml")
        Dim accountdetails As System.Xml.XmlNodeList =
accounts.SelectNodes("accounts/useraccountdetails")
        staffinitials =
accountdetails.Item(number).SelectSingleNode("staffinitials").InnerText()
    End Function
    Function department(ByVal number As Integer)
        Dim accounts As New System.Xml.XmlDocument
        accounts.Load("accountdetails.xml")
        Dim accountdetails As System.Xml.XmlNodeList =
accounts.SelectNodes("accounts/useraccountdetails")
        department =
accountdetails.Item(number).SelectSingleNode("department").InnerText()
    End Function
    Function datesubmitted(ByVal number As Integer)
        Dim accounts As New System.Xml.XmlDocument
        accounts.Load("accountdetails.xml")
        Dim accountdetails As System.Xml.XmlNodeList =
accounts.SelectNodes("accounts/useraccountdetails")
        datesubmitted =
accountdetails.Item(number).SelectSingleNode("datesubmitted").InnerText()
    End Function
    Function numberofcopies(ByVal number As Integer)
        Dim accounts As New System.Xml.XmlDocument
        accounts.Load("accountdetails.xml")
        Dim accountdetails As System.Xml.XmlNodeList =
accounts.SelectNodes("accounts/useraccountdetails")
        numberofcopies =
accountdetails.Item(number).SelectSingleNode("numberofcopies").InnerText()
    End Function
    Function totalprice(ByVal number As Integer)
        Dim accounts As New System.Xml.XmlDocument
        accounts.Load("accountdetails.xml")
        Dim accountdetails As System.Xml.XmlNodeList =
accounts.SelectNodes("accounts/useraccountdetails")
        totalprice =
accountdetails.Item(number).SelectSingleNode("totalprice").InnerText()
    End Function
    Function Length()
```

```
        Dim accounts As New System.Xml.XmlDocument
        accounts.Load("accountdetails.xml")
        Dim accountdetails As System.Xml.XmlNodeList =
accounts.SelectNodes("accounts/useraccountdetails")
        Return accountdetails.Count
    End Function
    Function saveaccountdetails(ByVal jobnumber As Integer, ByVal userid As Integer,
ByVal staffinitials As String, ByVal department As String, ByVal datesubmitted As
String, ByVal numberofcopies As Integer, ByVal totalprice As Integer)
        Dim useraccountdetailsxml = XDocument.Load("accountdetails.xml")
        Dim useraccountdetail As XElement = _
         <useraccountdetails>
             <jobnumber>[redacted] jobnumber [redacted]</jobnumber>
             <userid>[redacted] userid [redacted]</userid>
             <staffinitials>[redacted] staffinitials [redacted]</staffinitials>
             <department>[redacted] department [redacted]</department>
             <datesubmitted>[redacted] datesubmitted [redacted]</datesubmitted>
             <numberofcopies>[redacted] numberofcopies [redacted]</numberofcopies>
             <totalprice>[redacted] totalprice [redacted]</totalprice>
         </useraccountdetails>
        useraccountdetailsxml.<accounts>(0).Add(useraccountdetail)
        useraccountdetailsxml.Save("accountdetails.xml")
        Return True
    End Function
End Class
```

Add a user




```
Public Class Add_a_user
    Private Sub cancel_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cancel.Click
        Reprographics_usermenu.Show()
        Me.Close()
    End Sub

    Private Sub createnewuser_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles createnewuser.Click
```

```
        Dim getdetails As New gettinguserdetails
        Dim saveuserdetails As New userprofile
        Dim passwordcheck As Boolean
        Dim printcredits As New printcredits
        getdetails.userid = MaskedTextBox1.Text
        getdetails.department = MaskedTextBox2.Text

        getdetails.password = MaskedTextBox3.Text
        getdetails.confirmpassword = MaskedTextBox4.Text
        getdetails.securityquestion = TextBox1.Text
        getdetails.securityanswer = TextBox2.Text

        If getdetails.password = getdetails.confirmpassword Then
            passwordcheck = True
            printcredits.saveprintcredits(getdetails.userid, 1000)
            saveuserdetails.saveuser(getdetails.userid, getdetails.department,
getdetails.password, getdetails.confirmpassword, getdetails.securityquestion,
getdetails.securityanswer)
            MsgBox("User Successfully added", MsgBoxStyle.OkOnly, "Add a user")
        Else
            MsgBox("Password and confirm password are not the same, please enter them
again")
            passwordcheck = False
        End If
    End Sub

End Class
```

Queue (class)

```
Public Class addtoqueue
        Function addformtoqueue()
        Print_job_queue.Queue.Items.Clear()
        Dim jobitem As New printjobform
        Dim length As Integer
        Dim topofstack As String
        Dim number As Integer = 0
        Dim currentnumber As Integer = 0
        Dim nomoreswaps As Boolean
        Dim temporystoredvalue As String
        Dim childnumber As Integer
        Dim find As New findjob
        Dim initialisenumber As Integer
        length = jobitem.Length
        Dim sort(length) As String
        For initialisenumber = 1 To length
            sort(initialisenumber) = ""
        Next
        For number = 1 To length
            If jobitem.jobcomplete(number - 1) = "false" Then
                currentnumber = currentnumber + 1
                sort(currentnumber) = jobitem.requestdate(number - 1)
            End If
        Next
        Do
            nomoreswaps = True
            For value = 1 To length - 1
                If sort(value + 1) <> "" Then
                    If SortableDate(sort(value)) > SortableDate(sort(value + 1)) Then
                        nomoreswaps = False
                        temporystoredvalue = sort(value)
                        sort(value) = sort(value + 1)
                        sort(value + 1) = temporystoredvalue
                    End If
```

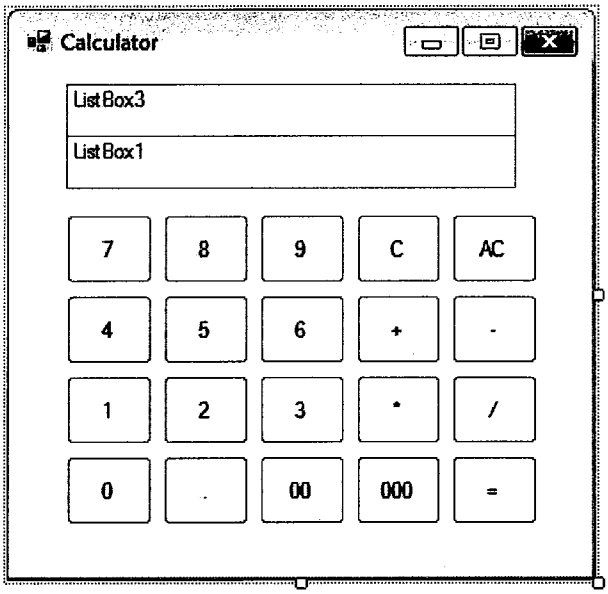
```
                End If
            Next
        Loop Until nomoreswaps = True
        For value = 1 To length
            If sort(value) <> "" Then
                childnumber = find.findrequestdate(sort(value))
                Print_job_queue.Queue.Items.Add(jobitem.jobnumber(childnumber) & "  "
& jobitem.requestdate(childnumber) & "  " & jobitem.staffinitials(childnumber))
            End If
        Next
        Try
            topofstack = Print_job_queue.Queue.Items.Item(0)
        Catch
            topofstack = 0
            MsgBox("no items in the queue", MsgBoxStyle.OkOnly, "No Jobs")
        End Try
        Return topofstack
    End Function
    Sub addaccountdetails()
        Dim accounts As New accounts
        Dim length As Integer
        Dim number As Integer = 0
        length = accounts.Length
        Reprographics_Accounts.ListBox1.Items.Clear()
        For number = 1 To length
            Reprographics_Accounts.ListBox1.Items.Add(accounts.userid(number - 1) & "
" & accounts.staffinitials(number - 1) & "                  " &
accounts.numberofcopies(number - 1) & "            " & "            " &
accounts.datesubmitted(number - 1) & "           £" & accounts.totalprice(number -
1))
        Next
    End Sub
    Sub addnewmessagetostack()
        Dim messageitem As New user_sent_message
        Dim length As Integer
        Dim number As Integer
        reprographics_check_messages.newmessagestack.Items.Clear()
        length = messageitem.length
        For number = 1 To length
            If messageitem.read(number - 1) = "false" Then

reprographics_check_messages.newmessagestack.Items.Add(messageitem.messagenumber(numbe
r - 1) & "   " & messageitem.userid(number - 1) & "      " & messageitem.name(number -
1) & "      " & messageitem.datesubmitted(number - 1) & "      " &
messageitem.topic(number - 1))
            End If
        Next

    End Sub
    Sub addpastmessagetoqueue()
        Dim messageitem As New user_sent_message
        Dim length As Integer
        Dim number As Integer = 0
        length = messageitem.length
        Reprographics_past_messages.pastmessagesstack.Items.Clear()
        For number = 1 To length
            If messageitem.read(number - 1) = "true" Then

Reprographics_past_messages.pastmessagesstack.Items.Add(messageitem.messagenumber(numb
er - 1) & "   " & messageitem.userid(number - 1) & "      " & messageitem.name(number
- 1) & "      " & messageitem.datesubmitted(number - 1) & "      " &
messageitem.topic(number - 1))
            End If
        Next
```

```
    End Sub
    Sub addtolistpastjobs()
        Dim submittedjobs As New printjobform
        Dim length As Integer
        Dim number As Integer = 0
        length = submittedjobs.Length
        View_print_forms.ListBox1.Items.Clear()
        For number = 1 To length
            If submittedjobs.userid(number - 1) = Login.username.Text Then
                View_print_forms.ListBox1.Items.Add(submittedjobs.jobnumber(number -
1) & "  " & submittedjobs.requestdate(number - 1) & "  " &
submittedjobs.staffinitials(number - 1))
            End If
        Next
    End Sub
    Sub adduserpastmessagetostack()
        Dim messageitem As New reprographics_sent_messages
        Dim findusernumber As New finduser
        Dim length As Integer
        Dim usernumber As Integer
        Dim number As Integer = 0
        usercheckpastmessages.messagepastmessagestack.Items.Clear()
        length = messageitem.Length
        usernumber =
findusernumber.finduserprofileformessagesentbyreprographics(Login.username.Text)
        For number = 1 To length
            If messageitem.read(number - 1) = "true" And Login.username.Text =
messageitem.userid(number - 1) Then

usercheckpastmessages.messagepastmessagestack.Items.Add(messageitem.messagenumber(numb
er - 1) & "   " & messageitem.userid(number - 1) & "      " & messageitem.name(number
- 1) & "      " & messageitem.datesubmitted(number - 1) & "      " &
messageitem.topic(number - 1))
            End If
        Next
    End Sub
    Sub addusernewmessagetostack(ByVal value)
        Dim messageitem As New reprographics_sent_messages
        Dim findusernumber As New finduser
        Dim doesuserhaveanewmessage As New does_user_have_a_new_message
        Dim length As Integer
        Dim usernumber As Integer
        Dim newmessage As Boolean
        length = messageitem.Length
        usercheckpastmessages.messagestack.Items.Clear()
        newmessage = doesuserhaveanewmessage.userprofilefound(value)
        If newmessage = True Then
            usernumber =
findusernumber.finduserprofileformessagesentbyreprographics(value)
            For number = 1 To length
                If messageitem.read(number - 1) = "false" And Login.username.Text =
messageitem.userid(number - 1) Then

usercheckpastmessages.messagestack.Items.Add(messageitem.messagenumber(number - 1) & "
" & messageitem.userid(number - 1) & "      " & messageitem.name(number - 1) & "
" & messageitem.datesubmitted(number - 1) & "      " & messageitem.topic(number - 1))
                End If
            Next
        End If
    End Sub
End Class
```

Calculate total price of print job

```
Public Class calculate_total_price_of_print_job
    Function calculatetotalprice(ByVal papertype As String, ByVal papersize As String,
ByVal printtype As String, ByVal papercolour As String, ByVal backtoback As Boolean,
ByVal collated As Boolean, ByVal stapled As Boolean, ByVal wirobinding As Boolean,
ByVal laminatea4 As Boolean, ByVal laminatea3 As Boolean, ByVal gluebound As Boolean,
ByVal transparencies As Boolean, ByVal twoholepunched As Boolean, ByVal folded As
Boolean, ByVal numberofcopies As Integer)
        Dim checkprice As New special_price_list
        calculatetotalprice = 0
        If printtype = "Black and White" Then
            calculatetotalprice = calculatetotalprice + checkprice.bwprint
        ElseIf printtype = "colour" Then
            calculatetotalprice = calculatetotalprice + checkprice.colourprint
        ElseIf backtoback = True Then
            calculatetotalprice = calculatetotalprice + checkprice.backtoback()
        ElseIf collated = True Then
            calculatetotalprice = calculatetotalprice + checkprice.collated()
        ElseIf stapled = True Then
            calculatetotalprice = calculatetotalprice + checkprice.stapled()
        ElseIf wirobinding = True Then
            calculatetotalprice = calculatetotalprice + checkprice.wirobinding
        ElseIf laminatea4 = True Then
            calculatetotalprice = calculatetotalprice + checkprice.laminatea4
        ElseIf laminatea3 = True Then
            calculatetotalprice = calculatetotalprice + checkprice.laminatea3
        ElseIf gluebound = True Then
            calculatetotalprice = calculatetotalprice + checkprice.gluebound
        ElseIf transparencies = True Then
            calculatetotalprice = calculatetotalprice + checkprice.transparencies
        ElseIf twoholepunched = True Then
            calculatetotalprice = calculatetotalprice + checkprice.twoholepunched
        ElseIf folded = True Then
            calculatetotalprice = calculatetotalprice + checkprice.folded()
        ElseIf papertype = "card" Then
            calculatetotalprice = calculatetotalprice + checkprice.card()
        ElseIf papertype = "normal" Then
            calculatetotalprice = calculatetotalprice + checkprice.stapled()
        ElseIf papersize = "a3" Then
            calculatetotalprice = calculatetotalprice + checkprice.papersizea3()
        ElseIf papersize = "a4" Then
            calculatetotalprice = calculatetotalprice + checkprice.papersizea4
        ElseIf papersize = "a5" Then
            calculatetotalprice = calculatetotalprice + checkprice.papersizea5
        ElseIf papercolour = "coloured" Then
            calculatetotalprice = calculatetotalprice + checkprice.papercolour
        ElseIf papercolour = "plain" Then
            calculatetotalprice = calculatetotalprice + checkprice.plainpaper
        End If
        calculatetotalprice = calculatetotalprice * numberofcopies
        Return calculatetotalprice
    End Function
End Class
```

Calculator



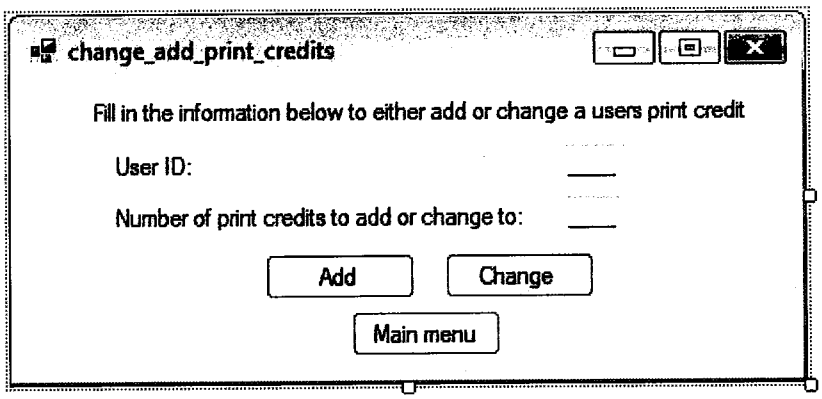

```
PublicClassForm1
Dim number AsString
Dim i AsInteger
Dim pointer AsInteger
Dim convertednumber AsDouble
Dim stack(20) AsInteger
Dim firstnumber AsSingle
Dim secondnumber AsSingle
Dim sum AsInteger
Dim add AsBoolean
Dim multiplication AsBoolean
Dim subtract AsBoolean
Dim divide AsBoolean
Dim Calculator AsNewCalculation
PrivateSub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        number = number + "7"
        update(number)
EndSub

PrivateSub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
        number = number + "8"
        update(number)
EndSub

PrivateSub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button3.Click
        number = number + "9"
        update(number)
EndSub

PrivateSub Button10_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button10.Click
        number = number + "4"
        update(number)
EndSub

PrivateSub Button9_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button9.Click
```

```
        number = number + "5"
        update(number)
EndSub

PrivateSub Button8_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button8.Click
        number = number + "6"
        update(number)
EndSub

PrivateSub Button15_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button15.Click
        number = number + "1"
        update(number)
EndSub

PrivateSub Button14_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button14.Click
        number = number + "2"
        update(number)
EndSub

PrivateSub Button13_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button13.Click
        number = number + "3"
        update(number)
EndSub

PrivateSub Button20_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button20.Click
        number = number + "0"
        update(number)
EndSub
Sub update2(ByVal convertednumber AsInteger)
        ListBox3.Items.Add(firstnumber)
        ListBox1.Items.Add(firstnumber)

EndSub

PrivateSub ListBox2_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        ListBox1.Items.Add(number)
EndSub
PrivateSub ListBox3_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ListBox3.SelectedIndexChanged
        ListBox1.Items.Add(firstnumber)
EndSub
Sub push(ByVal value AsDouble)
'pointer = pointer + 1
        secondnumber = firstnumber
        firstnumber = value
        stack(pointer) = convertednumber
        update2(convertednumber)
        ListBox1.Items.Clear()
        number = 0

EndSub

Sub update3(ByVal sum AsInteger)
        ListBox1.Items.Clear()
        ListBox1.Items.Add(sum)
EndSub
```

```
PrivateSub Button7_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button7.Click
        add = True
        convertednumber = Val(number)
        push(convertednumber)
EndSub

PrivateSub ListBox1_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ListBox1.SelectedIndexChanged
        ListBox1.Items.Add(sum)
EndSub
Sub update(ByVal number AsInteger)
        ListBox1.Items.Clear()
        ListBox1.Items.Add(number)

EndSub


PrivateSub Button6_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button6.Click
        subtract = True
        convertednumber = Val(number)
        push(convertednumber)
EndSub

PrivateSub Button12_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button12.Click
        multiplication = True
        convertednumber = Val(number)
        push(convertednumber)
EndSub

PrivateSub Button11_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button11.Click
        divide = True
        convertednumber = Val(number)
        push(convertednumber)
EndSub

PrivateSub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button11.Click
        number = 0
        ListBox1.Items.Clear()
'ListBox2.Items.Clear()
        ListBox3.Items.Clear()

EndSub

PrivateSub Button17_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button17.Click
        number = number + "000"
        update(number)
EndSub
PrivateSub Button18_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button18.Click
        number = number + "00"
        update(number)
EndSub

PrivateSub Button16_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button16.Click
        convertednumber = Val(number)
        push(convertednumber)
If add = TrueThen
```

```
            sum = Calculator.Add(firstnumber, secondnumber)
            update3(sum)
ElseIf subtract = TrueThen
            sum = Calculator.Subtract(secondnumber, firstnumber)
            update3(sum)
ElseIf divide = TrueThen
            sum = Calculator.Divide(secondnumber, firstnumber)
            update3(sum)
ElseIf multiplication = TrueThen
            sum = Calculator.Multiply(secondnumber, firstnumber)
            update3(sum)
EndIf

EndSub
EndClass
PublicClassCalculation
PublicFunction Add(ByVal firstnumber, ByVal secondnumber)
Return firstnumber + secondnumber
EndFunction
PublicFunction Subtract(ByVal firstnumber, ByVal secondnumber)
Return firstnumber - secondnumber
EndFunction
PublicFunction Multiply(ByVal firstnumber, ByVal secondnumber)
Return firstnumber * secondnumber
EndFunction
PublicFunction Divide(ByVal firstnumber, ByVal secondnumber)
Return firstnumber / secondnumber
EndFunction
EndClass
```

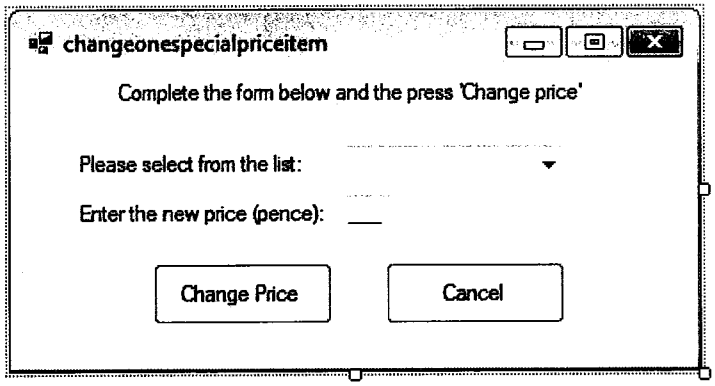
Change add print credits




```
Public Class change_add_print_credits

    Private Sub mainmenu_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mainmenu.Click
        Reprographics_usermenu.Show()
        Me.Close()
    End Sub

    Private Sub add_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles add.Click
        Dim changeadd As New printcredits
        Dim details As New getdetailsforprintcreditchange
        Dim findusernumber As New finduser
        Dim usernumber As Integer
        Dim currentnumberofcredits As Integer
```

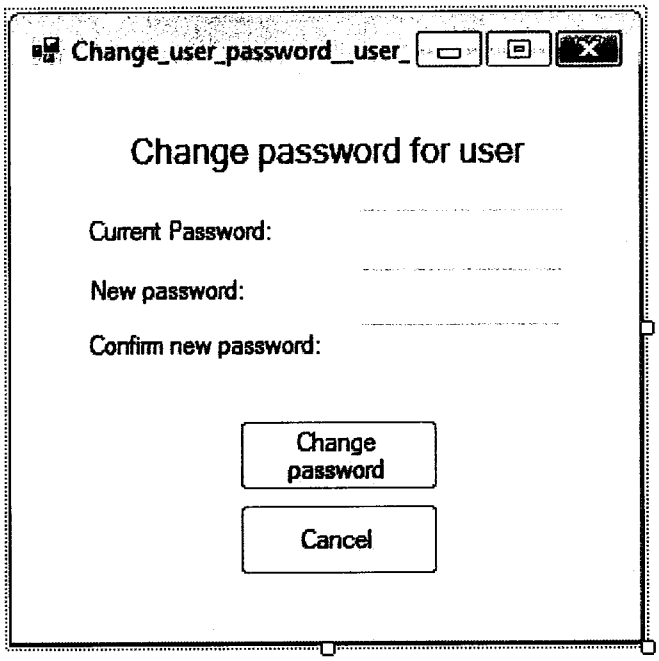
```
        Dim newprintcredits As Integer
        details.getuserid = userid.Text
        details.getnumberofprintcredits = numberofprintcredits.Text
        usernumber = findusernumber.finduserprofile(details.getuserid)
        currentnumberofcredits = changeadd.lookupprintcredits(usernumber)
        newprintcredits = currentnumberofcredits + details.getnumberofprintcredits
        changeadd.changeprintcredits(usernumber, newprintcredits)
    End Sub

    Private Sub change_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles change.Click
        Dim changeadd As New printcredits
        Dim details As New getdetailsforprintcreditchange
        Dim findusernumber As New finduser
        Dim usernumber As Integer
        details.getuserid = userid.Text
        details.getnumberofprintcredits = numberofprintcredits.Text
        usernumber = findusernumber.finduserprofile(details.getuserid)
        changeadd.changeprintcredits(usernumber, details.getnumberofprintcredits)
    End Sub
End Class
```

Change one special price item



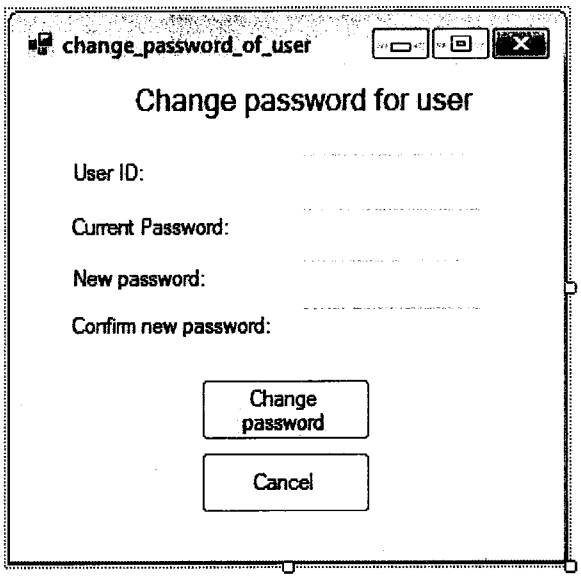

```
Public Class changeonespecialpriceitem

    Private Sub cancel_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cancel.Click
        Reprographics_Accounts.Show()
        Me.Close()
    End Sub

    Private Sub changeprice_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles changeprice.Click
        Dim getdetails As New get_change_price_details
        Dim savechanges As New special_price_list

        getdetails.getitemselected = speicalpriceitemselected.Text
        getdetails.getnewprice = newspecialitemprice.Text
        savechanges.changespecialprice(getdetails.getitemselected,
getdetails.getnewprice)
        MsgBox("Price change successfully")
    End Sub
End Class
```

Change password of user



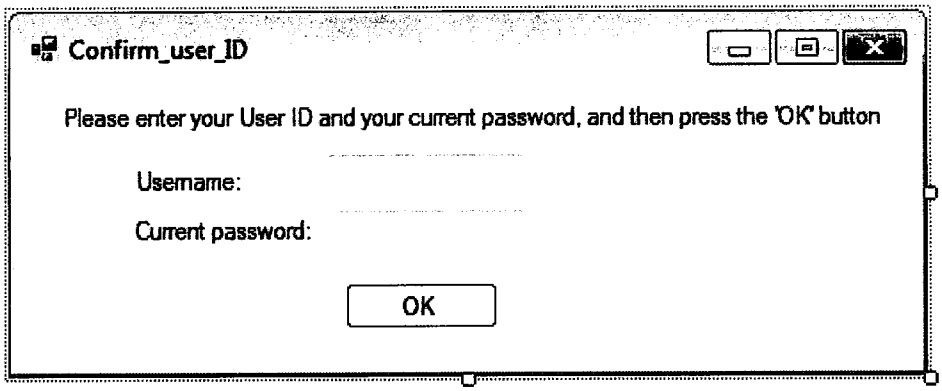

```
Public Class change_username_password_of_user

    Private Sub cancel_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cancel.Click
        Reprographics_usermenu.Show()
        Me.Close()
    End Sub
    Private Sub changepassword_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles changepassword.Click

        Dim userid As String
        Dim finduserdetails As New finduser
        Dim entereduserid As New getenteredlogindetails
        Dim currentpasswordentered As String
        Dim newpassword As String
        Dim newconfirmedpassword As String
        Dim usernumber As Integer
        Dim userprofile As New userprofile

        userid = Confirm_user_ID.UserID.Text & Confirm_user_ID.currentpassword.Text
        usernumber = finduserdetails.findusersequrityquestion(userid)
        currentpasswordentered = MaskedTextBox2.Text
        newpassword = MaskedTextBox3.Text
        newconfirmedpassword = MaskedTextBox4.Text
        userprofile.changepassword(usernumber, newpassword)
    End Sub

End Class
```

Change password of user



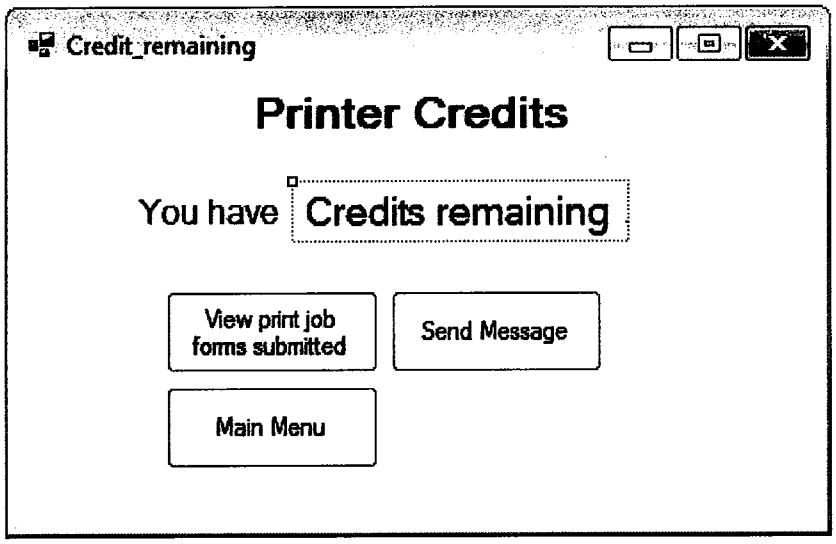

```
Public Class change_username_password_of_user

    Private Sub cancel_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cancel.Click
        Reprographics_usermenu.Show()
        Me.Close()
    End Sub
    Private Sub changepassword_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles changepassword.Click

        Dim userid As String
        Dim finduserdetails As New finduser
        Dim entereduserid As New getenteredlogindetails
        Dim currentpasswordentered As String
        Dim newpassword As String
        Dim newconfirmedpassword As String
        Dim usernumber As Integer
        Dim userprofile As New userprofile

        userid = Confirm_user_ID.UserID.Text & Confirm_user_ID.currentpassword.Text
        usernumber = finduserdetails.findusersequrityquestion(userid)
        currentpasswordentered = MaskedTextBox2.Text
        newpassword = MaskedTextBox3.Text
        newconfirmedpassword = MaskedTextBox4.Text
        userprofile.changepassword(usernumber, newpassword)
    End Sub

End Class
```

Confirm user ID and password




```
Public Class Confirm_user_ID

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim userfound As New finduser
        Dim checkuser As Boolean
        Dim entereduserid As New getenteredlogindetails
        Dim secondstageofforgottenpassword As New forgotten_password_final_stage
        Dim answer As String
        entereduserid.username = UserID.Text
        entereduserid.password = currentpassword.Text
        answer = entereduserid.username & entereduserid.password
        checkuser = userfound.userpasswordfound(answer)
        If checkuser = True Then
            MsgBox("User ID verified, please enter your security answer",
MsgBoxStyle.OkOnly, "UserID verification")
            Change_user_password__user_.Show()
            Me.Close()
        ElseIf checkuser = False Then
            MsgBox("Invalid UserID, please enter it again or if you do not have an
UserID see the reprographics", MsgBoxStyle.OkOnly, "UserID verification")
        End If
    End Sub
End Class
```

Credit remaining




```
Public Class Credit_remaining

    Private Sub mainmenu_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mainmenu.Click
        User_Menu.Show()
        Me.Close()
    End Sub

    Private Sub sendmessage_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles sendmessage.Click
        Send_Message.Show()
        Me.Close()
    End Sub

    Private Sub Viewprintjobformssubmitted_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles Viewprintjobformssubmitted.Click
        View_print_forms.Show()
        Me.Close()
    End Sub

    Private Sub Credit_remaining_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim printcredits As New printcredits
        Dim findusernumber As New finduser
        Dim usernumber As Integer
        usernumber = findusernumber.finduserprofile(Login.username.Text)
        creditsremaining.Text = printcredits.printcredits(usernumber)
    End Sub
End Class
```

Does user have a new message

```
Public Class does_user_have_a_new_message
    Function userprofilefound(ByVal value)
        Dim number As Integer = 0
        Dim checkprofiledata As New reprographics_sent_messages
        Dim profilefound As Boolean
        Do
            If value = (checkprofiledata.userid(number)) Then
                profilefound = True
            Else
```

```
                profilefound = False
                number = number + 1
            End If
        Loop Until number = checkprofiledata.Length Or profilefound = True
        Return profilefound
    End Function
End Class
```

Find job

```
Public Class findjob
    Function findrightjob(ByVal find As String)
        Dim jobnumber As New printjobform
        Dim length As Integer
        Dim foundjobnumber As Integer
        Dim jobfound As Boolean
        Dim number As Integer = 0
        length = jobnumber.Length

        Do
            If find = (jobnumber.jobnumber(number) & "  " &
jobnumber.requestdate(number) & "  " & jobnumber.staffinitials(number)) Then
                jobfound = True
                foundjobnumber = number

            Else
                jobfound = False
                number = number + 1
            End If
        Loop Until number = length Or jobfound = True
        findrightjob = foundjobnumber
        Return findrightjob
    End Function
    Function findrequestdate(ByVal find As String)
        Dim jobnumber As New printjobform
        Dim length As Integer
        Dim foundjobnumber As Integer
        Dim jobfound As Boolean
        Dim number As Integer = 0
        length = jobnumber.Length

        Do
            If find = jobnumber.requestdate(number) Then
                jobfound = True
                foundjobnumber = number

            Else
                jobfound = False
                number = number + 1
            End If
        Loop Until number = length Or jobfound = True
        findrequestdate = foundjobnumber
        Return findrequestdate
    End Function

    Function findrightaccountuseriddetails(ByVal find As String)
        Dim jobnumber As New accounts
        Dim length As Integer
        Dim foundjobnumber As Integer
        Dim detailsfound As Boolean
        Dim number As Integer = 0
        length = jobnumber.Length
```

```
        Do
            If find = jobnumber.userid(number) Then
                detailsfound = True
                foundjobnumber = number

            Else
                detailsfound = False
                number = number + 1
            End If
        Loop Until number = length Or detailsfound = True
        findrightaccountuseriddetails = foundjobnumber
        Return findrightaccountuseriddetails


    End Function
    Function findrightaccountnumberofcopiesdetails(ByVal find As String)
        Dim jobnumber As New accounts
        Dim length As Integer
        Dim foundjobnumber As Integer
        Dim detailsfound As Boolean
        Dim number As Integer
        length = jobnumber.Length
        number = 0
        Do
            If find = jobnumber.numberofcopies(number) Then
                detailsfound = True
                foundjobnumber = number

            Else
                detailsfound = False
                number = number + 1
            End If
        Loop Until number = length Or detailsfound = True
        findrightaccountnumberofcopiesdetails = foundjobnumber
        Return findrightaccountnumberofcopiesdetails


    End Function
    Function findrightaccountdatesubmitteddetails(ByVal find As String)
        Dim jobnumber As New accounts
        Dim length As Integer
        Dim foundjobnumber As Integer
        Dim detailsfound As Boolean
        Dim number As Integer
        length = jobnumber.Length
        number = 0
        Do
            If find = jobnumber.datesubmitted(number) Then
                detailsfound = True
                foundjobnumber = number

            Else
                detailsfound = False
                number = number + 1
            End If
        Loop Until number = length Or detailsfound = True
        findrightaccountdatesubmitteddetails = foundjobnumber
        Return findrightaccountdatesubmitteddetails
    End Function
    Function findrightaccounttotalcostdetails(ByVal find As String)
        Dim jobnumber As New accounts
        Dim length As Integer
        Dim foundjobnumber As Integer
        Dim detailsfound As Boolean
```

```
        Dim number As Integer
        length = jobnumber.Length
        number = 0
        Do
            If find = jobnumber.totalprice(number) Then
                detailsfound = True
                foundjobnumber = number

            Else
                detailsfound = False
                number = number + 1
            End If
        Loop Until number = length Or detailsfound = True
        findrightaccounttotalcostdetails = foundjobnumber
        Return findrightaccounttotalcostdetails
    End Function
    Function findrightmessage(ByVal find As String)
        Dim messagenumber As New user_sent_message
        Dim length As Integer
        Dim foundmessagenumber As Integer
        Dim messagenumberfound As Boolean
        Dim number As Integer
        length = messagenumber.Length
        number = 0
        Do
            If find = (messagenumber.messagenumber(number) & "    " &
messagenumber.userid(number) & "      " & messagenumber.name(number) & "      " &
messagenumber.datesubmitted(number) & "      " & messagenumber.topic(number)) Then
                messagenumberfound = True
                foundmessagenumber = number

            Else
                messagenumberfound = False
                number = number + 1
            End If
        Loop Until number = length Or messagenumberfound = True
        findrightmessage = foundmessagenumber
        Return findrightmessage
    End Function
    Function findrightreprographicsmessage(ByVal find As String)
        Dim messagenumber As New reprographics_sent_messages
        Dim length As Integer
        Dim foundmessagenumber As Integer
        Dim messagenumberfound As Boolean
        Dim number As Integer
        length = messagenumber.length
        number = 0
        Do
            If find = (messagenumber.messagenumber(number) & "    " &
messagenumber.userid(number) & "      " & messagenumber.name(number) & "      " &
messagenumber.datesubmitted(number) & "      " & messagenumber.topic(number)) Then
                messagenumberfound = True
                foundmessagenumber = number

            Else
                messagenumberfound = False
                number = number + 1
            End If
        Loop Until number = length Or messagenumberfound = True
        findrightreprographicsmessage = foundmessagenumber
        Return findrightreprographicsmessage
    End Function
    Function findrightmessageinuserlogin(ByVal find As String)
        Dim messageitem As New reprographics_sent_messages
```

```
        Dim length As Integer
        Dim foundmessagenumber As Integer
        Dim messagenumberfound As Boolean
        Dim number As Integer
        length = messageitem.Length
        number = 0
        Do
            If find = (messageitem.messagenumber(number) & "    " &
messageitem.userid(number) & "      " & messageitem.name(number) & "      " &
messageitem.datesubmitted(number) & "      " & messageitem.topic(number)) Then
                messagenumberfound = True
                foundmessagenumber = number

            Else
                messagenumberfound = False
                number = number + 1
            End If
        Loop Until number = length Or messagenumberfound = True
        findrightmessageinuserlogin = foundmessagenumber
        Return findrightmessageinuserlogin
    End Function
End Class
```

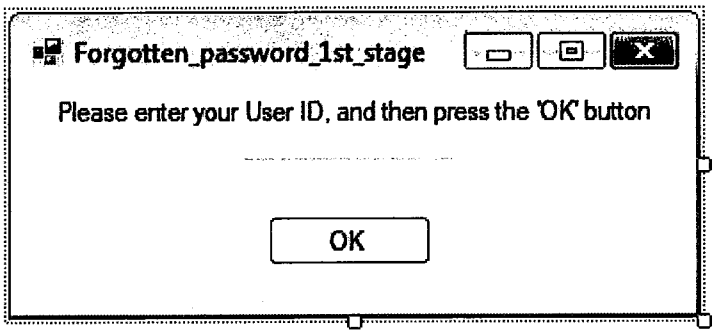
Find user

```
Public Class finduser
    Function findrightuser(ByVal username As Integer, ByVal password As String)
        Dim usernumber As New userprofile
        Dim length As Integer
        Dim number As Integer
        Dim validuser As Boolean
        Dim foundusernumber As Integer
        Dim loginvalid As Boolean
        Dim a As String
        length = usernumber.Length
        a = username & password
        validuser = userfound(a)
        Do
            If a = (usernumber.userid(number) & usernumber.password(number)) Then
                loginvalid = True
                foundusernumber = number

            Else
                loginvalid = False
                number = number + 1
            End If
        Loop Until number = length Or loginvalid = True

        findrightuser = foundusernumber + 1
        Return findrightuser
    End Function

    Function userpasswordfound(ByVal value)
        Dim number As Integer = 0
        Dim checkuserdata As New userprofile

        Do
            If value = (checkuserdata.userid(number) & checkuserdata.password(number))
Then
                userpasswordfound = True
            Else
                userpasswordfound = False
```

```
            number = number + 1
        End If
    Loop Until number = checkuserdata.Length Or userpasswordfound = True
    Return userpasswordfound
End Function
Function findusersequrityquestion(ByVal userid As String)
    Dim usernumber As New userprofile
    Dim length As Integer
    Dim number As Integer
    Dim foundusernumber As Integer
    Dim loginvalid As Boolean
    Dim a As String
    a = userid
    length = usernumber.Length
    number = 0
    Do
        If a = usernumber.userid(number) Then
            loginvalid = True
            foundusernumber = number

        Else
            loginvalid = False
            number = number + 1

        End If
    Loop Until number = length Or loginvalid = True

    findusersequrityquestion = foundusernumber
    Return findusersequrityquestion
End Function

Function userfound(ByVal value)
    Dim number As Integer
    Dim checkuserdata As New userprofile
    number = 0
    Do
        If value = (checkuserdata.userid(number)) Then
            userfound = True
        Else
            userfound = False
            number = number + 1
        End If
    Loop Until number = checkuserdata.Length Or userfound = True
    Return userfound
End Function
Function finduserprofile(ByVal value)
    Dim number As Integer
    Dim checkprofiledata As New userprofile
    Dim profilefound As Boolean
    number = 0
    Do
        If value = (checkprofiledata.userid(number)) Then
            profilefound = True
        Else
            profilefound = False
            number = number + 1
        End If
    Loop Until number = checkprofiledata.Length - 1 Or profilefound = True
    Return number
End Function
Function finduserprofileformessagesentbyreprographics(ByVal value)
    Dim number As Integer
    Dim checkprofiledata As New reprographics_sent_messages
    Dim profilefound As Boolean
```

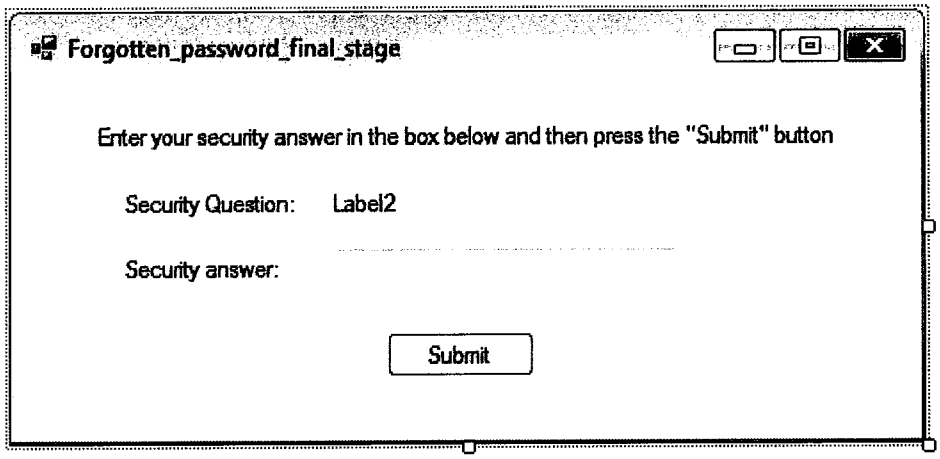
```
        number = 0
        Do
            If value = (checkprofiledata.userid(number)) Then
                profilefound = True
            Else
                profilefound = False
                number = number + 1
            End If
        Loop Until number = checkprofiledata.Length Or profilefound = True
        Return number
    End Function
    Function finddepartment(ByVal value) As String
        Dim checkprofiledata As New userprofile
        finddepartment = checkprofiledata.department(value)
    End Function
End Class
```

Forgotten password 1st stage




```
Public Class Forgotten_password_1st_stage

    Private Sub ok_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ok.Click
        Dim userfound As New finduser
        Dim checkuser As Boolean
        Dim useridentered As Integer
        Dim secondstageofforgottenpassword As New Forgotten_password_final_stage

        useridentered = UserID.Text
        checkuser = userfound.userfound(useridentered)
        If checkuser = True Then
            MsgBox("User ID verified, please enter your security answer",
MsgBoxStyle.OkOnly, "UserID verification")
            Forgotten_password_final_stage.Show()
            Me.Visible = False

        ElseIf checkuser = False Then
            MsgBox("Invalid UserID, please enter it again or if you do not have an
UserID see the reprographics", MsgBoxStyle.OkOnly, "UserID verification")
        End If

    End Sub
End Class
```

Forgotten password final stage




```
Public Class Forgotten_password_final_stage

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim securityanswered As New getsecurityanswer
        Dim userid As Integer
        Dim finduserdetails As New finduser
        Dim entereduserid As New getenteredlogindetails
        Dim usernumber As Integer
        Dim getuserdata As New userprofile
        securityanswered.securityanswer = answer.Text
        userid = Forgotten_password_1st_stage.UserID.Text
        usernumber = finduserdetails.findusersequrityquestion(userid)
        If answer.Text = getuserdata.securityanswer(usernumber) Then
            MsgBox("Your password is: " & getuserdata.password(usernumber))
            Login.Show()
            Me.Close()
        Else
            MsgBox("Wrong security answer, please try again or if you have forgotten
the answer to your security question see the reprographic for further assistance")

        End If
    End Sub

    Private Sub Forgotten_password_final_stage_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        Dim finduserdetails As New finduser
        Dim getuserdata As New userprofile
        Dim usernumber As Integer
        Dim entereduserid As New getenteredlogindetails
        Dim userid As Integer

        userid = entereduserid.username

        usernumber = finduserdetails.findusersequrityquestion(userid)
        Label2.Text = getuserdata.securityquestion(usernumber)
    End Sub
End Class
```

Get change price details

```
Public Class get_change_price_details
    Private itemselected As String
    Private newprice As Integer
    Public Property getitemselected As String
        Get
            Return itemselected
        End Get
        Set(ByVal value As String)
            itemselected = value
        End Set
    End Property
    Public Property getnewprice As Integer
        Get
            Return newprice
        End Get
        Set(ByVal value As Integer)
            newprice = value
        End Set
    End Property
End Class
```

Get user details

```
Public Class gettinguserdetails
    Private item1 As Integer
    Private item2 As String
    Private item3 As String
    Private item4 As String
    Private item5 As String
    Private item6 As String
    Public Property userid As Integer
        Get
            Return item1
        End Get
        Set(ByVal value As Integer)
            item1 = value
        End Set
    End Property
    Public Property department As String
        Get
            Return item2
        End Get
        Set(ByVal value As String)
            item2 = value
        End Set
    End Property
    Public Property password As String
        Get
            Return item3
        End Get
        Set(ByVal value As String)
            item3 = value
        End Set
    End Property
    Public Property confirmpassword As String
        Get
            Return item4
```

```
        End Get
        Set(ByVal value As String)
            item4 = value
        End Set
    End Property
    Public Property securityquestion As String
        Get
            Return item5
        End Get
        Set(ByVal value As String)
            item5 = value
        End Set
    End Property
    Public Property securityanswer As String
        Get
            Return item6
        End Get
        Set(ByVal value As String)
            item6 = value
        End Set
    End Property
End Class
```

Get details for print credit change

```
Public Class getdetailsforprintcreditchange
    Private userid As Integer
    Private numberofprintcredits As Integer
    Public Property getuserid As Integer
        Get
            Return userid
        End Get
        Set(ByVal value As Integer)
            userid = value
        End Set
    End Property
    Public Property getnumberofprintcredits As Integer
        Get
            Return numberofprintcredits
        End Get
        Set(ByVal value As Integer)
            numberofprintcredits = value
        End Set
    End Property
End Class
```

Get entered login details

```
Public Class getenteredlogindetails
    Private item1 As Integer
    Private item2 As String

    Public Property username As Integer
        Get
            Return item1
        End Get
        Set(ByVal value As Integer)
            item1 = value
        End Set
    End Property
    Public Property password As String
```

```
        Get
            Return item2
        End Get
        Set(ByVal value As String)
            item2 = value
        End Set
    End Property
End Class
```

Get message details

```
Public Class getmessagedetails
    Private name As String
    Private userid As Integer
    Private todaysdate As String
    Private topic As String
    Private message As String
    Public Property getname As String
        Get
            Return name
        End Get
        Set(ByVal value As String)
            name = value
        End Set
    End Property
    Public Property getuserid As Integer
        Get
            Return userid
        End Get
        Set(ByVal value As Integer)
            userid = value
        End Set
    End Property
    Public Property getdatesubmitted As String
        Get
            Return todaysdate
        End Get
        Set(ByVal value As String)
            todaysdate = value
        End Set
    End Property
    Public Property gettopic As String
        Get
            Return topic
        End Get
        Set(ByVal value As String)
            topic = value
        End Set
    End Property
    Public Property getmessage As String
        Get
            Return message
        End Get
        Set(ByVal value As String)
            message = value
        End Set
    End Property
End Class
```

Get new prices on everything

```
Public Class getnewpricesoneverything
    Private papersizea3 As String
    Private papersizea4 As String
    Private papersizea5 As String
    Private normal As String
    Private card As String
    Private bwprint As String
    Private colourprint As String
    Private plainpaper As String
    Private colourpaper As String
    Private backtoback As String
    Private collated As String
    Private stapled As String
    Private wirobinding As String
    Private laminatea4 As String
    Private laminatea3 As String
    Private gluebound As String
    Private transparencies As String
    Private twoholepunched As String
    Private folded As String
    Public Property getpapersizea3 As String
        Get
            Return papersizea3
        End Get
        Set(ByVal value As String)
            papersizea3 = value
        End Set
    End Property
    Public Property getpapersizea4 As String
        Get
            Return papersizea4
        End Get
        Set(ByVal value As String)
            papersizea4 = value
        End Set
    End Property
    Public Property getpapersizea5 As String
        Get
            Return papersizea5
        End Get
        Set(ByVal value As String)
            papersizea5 = value
        End Set
    End Property
    Public Property getnormal As String
        Get
            Return normal
        End Get
        Set(ByVal value As String)
            normal = value
        End Set
    End Property
    Public Property getcard As String
        Get
            Return card
        End Get
        Set(ByVal value As String)
            card = value
        End Set
    End Property
    Public Property getbwprint As String
```

```
        Get
            Return bwprint
        End Get
        Set(ByVal value As String)
            bwprint = value
        End Set
    End Property
    Public Property getcolouredprint As String
        Get
            Return colourprint
        End Get
        Set(ByVal value As String)
            colourprint = value
        End Set
    End Property
    Public Property getplainpaper As String
        Get
            Return plainpaper
        End Get
        Set(ByVal value As String)
            plainpaper = value
        End Set
    End Property
    Public Property getcolourpaper As String
        Get
            Return colourpaper
        End Get
        Set(ByVal value As String)
            colourpaper = value
        End Set
    End Property
    Public Property getbacktoback As String
        Get
            Return backtoback
        End Get
        Set(ByVal value As String)
            backtoback = value
        End Set
    End Property
    Public Property getcollated As String
        Get
            Return collated
        End Get
        Set(ByVal value As String)
            collated = value
        End Set
    End Property
    Public Property getstapled As String
        Get
            Return stapled
        End Get
        Set(ByVal value As String)
            stapled = value
        End Set
    End Property
    Public Property getwirobinding As String
        Get
            Return wirobinding
        End Get
        Set(ByVal value As String)
            wirobinding = value
        End Set
    End Property
    Public Property getlaminatea4 As String
```

```
        Get
            Return laminatea4
        End Get
        Set(ByVal value As String)
            laminatea4 = value
        End Set
    End Property
    Public Property getlaminatea3 As String
        Get
            Return laminatea3
        End Get
        Set(ByVal value As String)
            laminatea3 = value
        End Set
    End Property
    Public Property getgluebound As String
        Get
            Return gluebound
        End Get
        Set(ByVal value As String)
            gluebound = value
        End Set
    End Property
    Public Property gettransparencies As String
        Get
            Return transparencies
        End Get
        Set(ByVal value As String)
            transparencies = value
        End Set
    End Property
    Public Property gettwoholepunched As String
        Get
            Return twoholepunched
        End Get
        Set(ByVal value As String)
            twoholepunched = value
        End Set
    End Property
    Public Property getfolded As String
        Get
            Return folded
        End Get
        Set(ByVal value As String)
            folded = value
        End Set
    End Property
End Class
```

Get print job form details

```
Public Class getprintjobformdetails
    Private item1 As String
    Private item2 As Integer
    Private item3 As String
    Private item4 As String
    Private item5 As Integer
    Private item6 As String
    Private item7 As Boolean
    Private item8 As Boolean
    Private item9 As Boolean
```

```
    Private item10 As Boolean

    Private item11 As Boolean
    Private item12 As Boolean
    Private item13 As Boolean
    Private item14 As Boolean
    Private item15 As Boolean
    Private item16 As Boolean
    Private item17 As String
    Private item18 As String
    Private item19 As String


    Public Property staffinitials() As String
        Get
            Return item1
        End Get
        Set(ByVal value As String)
            item1 = value
        End Set
    End Property
    Public Property userid() As Integer
        Get
            Return item2
        End Get
        Set(ByVal value As Integer)
            item2 = value
        End Set
    End Property
    Public Property requestdate() As String
        Get
            Return item3
        End Get
        Set(ByVal value As String)
            item3 = value
        End Set
    End Property
    Public Property copysuppliedas() As String
        Get
            Return item4
        End Get
        Set(ByVal value As String)
            item4 = value
        End Set
    End Property
    Public Property numberofcopies() As Integer
        Get
            Return item5
        End Get
        Set(ByVal value As Integer)
            item5 = value
        End Set
    End Property
    Public Property papercolour() As String
        Get
            Return item6
        End Get
        Set(ByVal value As String)
            item6 = value
        End Set
    End Property
    Public Property backtoback() As Boolean
        Get
            Return item7
```

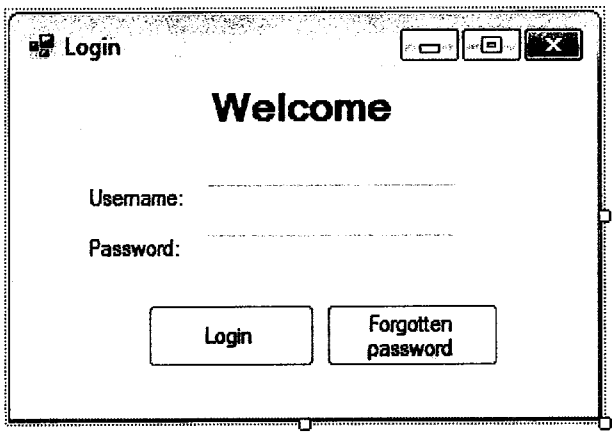
```
        End Get
        Set(ByVal value As Boolean)
            item7 = value
        End Set
    End Property
    Public Property collated() As Boolean
        Get
            Return item8
        End Get
        Set(ByVal value As Boolean)
            item8 = value
        End Set
    End Property
    Public Property stapled() As Boolean
        Get
            Return item9
        End Get
        Set(ByVal value As Boolean)
            item9 = value
        End Set
    End Property
    Public Property wirobinding() As Boolean
        Get
            Return item10
        End Get
        Set(ByVal value As Boolean)
            item10 = value
        End Set
    End Property
    Public Property laminate() As Boolean
        Get
            Return item11
        End Get
        Set(ByVal value As Boolean)
            item11 = value
        End Set
    End Property
    Public Property gluebound() As Boolean
        Get
            Return item12
        End Get
        Set(ByVal value As Boolean)
            item12 = value
        End Set
    End Property
    Public Property transparencies() As Boolean
        Get
            Return item13
        End Get
        Set(ByVal value As Boolean)
            item13 = value
        End Set
    End Property
    Public Property twoholepunched() As Boolean
        Get
            Return item14
        End Get
        Set(ByVal value As Boolean)
            item14 = value
        End Set
    End Property
    Public Property folded() As Boolean
        Get
            Return item15
```

```
        End Get
        Set(ByVal value As Boolean)
            item15 = value
        End Set
    End Property
    Public Property proofrequired() As Boolean
        Get
            Return item16
        End Get
        Set(ByVal value As Boolean)
            item16 = value
        End Set
    End Property
    Public Property papertype As String
        Get
            Return item17
        End Get
        Set(ByVal value As String)
            item17 = value
        End Set
    End Property
    Public Property papersize As String
        Get
            Return item18
        End Get
        Set(ByVal value As String)
            item18 = value
        End Set
    End Property
    Public Property printtype As String
        Get
            Return item19
        End Get
        Set(ByVal value As String)
            item19 = value
        End Set
    End Property
End Class
```

Get security answer

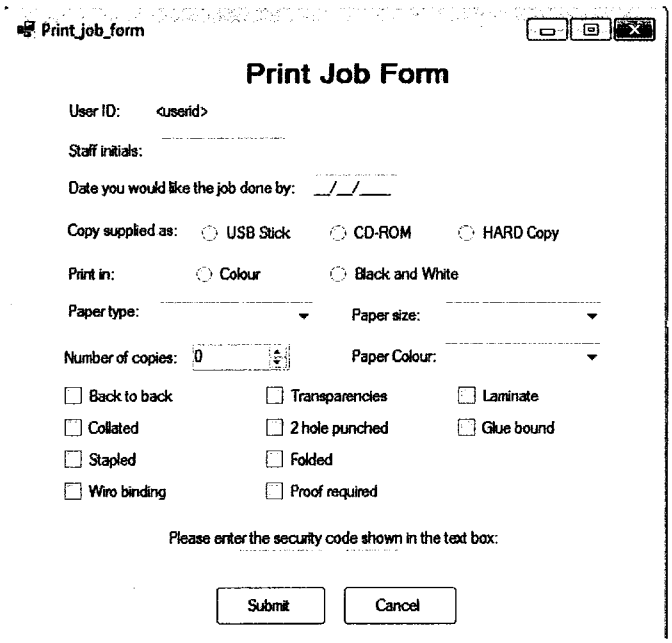
```
Public Class getsecurityanswer
    Private item1 As String

    Public Property securityanswer As String
        Get
            Return item1
        End Get
        Set(ByVal value As String)
            item1 = value
        End Set
    End Property
End Class
```

Login




```
Public Class Login

    Private Sub loginbutton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles loginbutton.Click
        Dim loggingdetails As New getenteredlogindetails
        Dim validlogin As New finduser
        Dim usernamepassword As String
        Dim checkuser As Boolean
        loggingdetails.username = username.Text
        loggingdetails.password = password.Text
        usernamepassword = loggingdetails.username & loggingdetails.password

        checkuser = validlogin.userpasswordfound(usernamepassword)

        If checkuser = True Then
            If loggingdetails.username = 8000 Then
                MsgBox("Welcome to the system")
                Reprographics_usermenu.Show()
                Me.Visible = False
            Else
                MsgBox("Welcome to the system")
                User_Menu.Show()
                Me.Visible = False
            End If
        ElseIf checkuser = False Then
            MsgBox("Invalid username/password")
        End If


    End Sub

    Private Sub forgottenpasswordbutton_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles forgottenpasswordbutton.Click
        Forgotten_password_1st_stage.Show()
        Me.Visible = False
    End Sub
End Class
```

Print job form



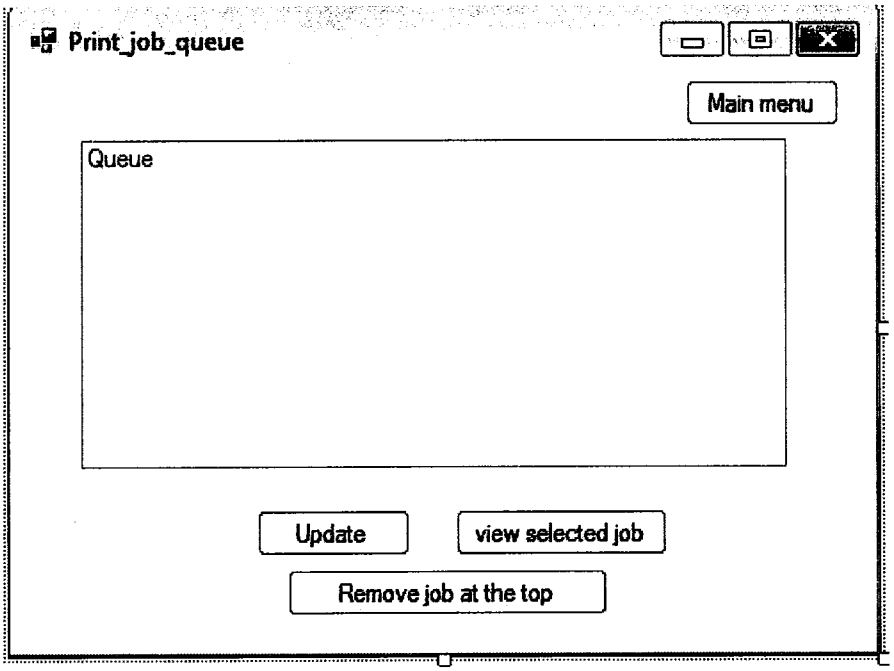

```
Public Class Print_job_form
    Dim newform As New getprintjobformdetails
    Dim job As New printjobform
    Private Sub submitted_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles submitted.Click
        Dim totalprice As Integer
        Dim gettotalprice As New calculate_total_price_of_print_job
        Dim accountdetails As New accounts
        Dim confirm As New printcredits
        Dim canjobproceed As Boolean = False
        Dim laminatea3 As Boolean
        Dim laminatea4 As Boolean
        Dim check As New printjobform
        Dim jobnumber As Integer = check.Length
        Dim checkuser As New finduser
        Dim uservalid As Boolean = False
        Dim sendprintjobform As Boolean = True
        Dim department As String
        Dim usernumber As Integer
        Dim user As New finduser

        uservalid = checkuser.userfound(userid.Text)
        usernumber = user.finduserprofile(userid.Text)
        department = user.finddepartment(usernumber)

        If securitycodeentered.Text = securitycode.Text And uservalid = True Then
            jobnumber = jobnumber + 1
            If userid.Text <> "" Then
                newform.userid = userid.Text
            Else
                sendprintjobform = False
                MsgBox("userid not entered")
            End If
            If TextBox1.Text <> "" Then
                newform.staffinitials = TextBox1.Text
            Else
```

```
                sendprintjobform = False
                MsgBox("staffinitials not entered")
            End If
            If MaskedTextBox1.Text <> "" Then
                newform.requestdate = MaskedTextBox1.Text
            Else
                sendprintjobform = False
                MsgBox("Request date not entered")
            End If
            If cdrom.Checked = True Then
                newform.copysuppliedas = cdrom.Text
            ElseIf usbstick.Checked = True Then
                newform.copysuppliedas = usbstick.Text
            ElseIf hardcopy.Checked = True Then
                newform.copysuppliedas = hardcopy.Text
            ElseIf cdrom.Checked = False And usbstick.Checked = False And
hardcopy.Checked = False Then
                sendprintjobform = False
                MsgBox("copysupplied not entered")
            End If
            If colour.Checked = True Then
                newform.printtype = colour.Text
            ElseIf bandw.Checked = True Then
                newform.printtype = bandw.Text
            ElseIf colour.Checked = False And bandw.Checked = False Then
                sendprintjobform = False
                MsgBox("print type not entered")
            End If
            If ComboBox2.Text <> "" Then
                newform.papertype = ComboBox2.Text
            Else
                sendprintjobform = False
            End If
            If ComboBox3.Text <> "" Then
                newform.papersize = ComboBox3.Text
            Else
                sendprintjobform = False
                MsgBox("paper size not entered")
            End If

            If NumericUpDown1.Value <= 700 Then
                newform.numberofcopies = NumericUpDown1.Text
                canjobproceed = confirm.updateprintcredits(newform.userid,
newform.numberofcopies)
            Else
                MsgBox("You have gone over the maximum limit which is 700 copies,
please input number 700 and below")
            End If
            If ComboBox1.Text <> "" Then
                newform.papercolour = ComboBox1.Text
            Else
                sendprintjobform = False
                MsgBox("paper colour not entered")
            End If
            newform.backtoback = CheckBox1.Checked
            newform.collated = CheckBox2.Checked
            newform.stapled = CheckBox3.Checked
            newform.wirobinding = CheckBox4.Checked
            newform.laminate = CheckBox5.Checked
            If newform.laminate = True And newform.papersize = "a3" Then
                laminatea3 = True
            ElseIf newform.laminate = True And newform.papersize = "a4" Then
                laminatea4 = True
            End If
```

```
            newform.gluebound = CheckBox6.Checked
            newform.transparencies = CheckBox12.Checked
            newform.twoholepunched = CheckBox11.Checked
            newform.folded = CheckBox10.Checked
            newform.proofrequired = CheckBox9.Checked
            If canjobproceed = True And sendprintjobform = True Then
                totalprice = gettotalprice.calculatetotalprice(newform.papertype,
newform.papersize, newform.printtype, newform.papercolour, newform.backtoback,
newform.collated, newform.stapled, newform.wirobinding, laminatea4, laminatea3,
newform.gluebound, newform.transparencies, newform.twoholepunched, newform.folded,
newform.numberofcopies)
                job.Add(jobnumber, newform.userid, newform.staffinitials,
newform.requestdate, newform.copysuppliedas, newform.printtype, newform.papertype,
newform.papersize, newform.numberofcopies, newform.papercolour, newform.backtoback,
newform.collated, newform.stapled, newform.wirobinding, newform.laminate,
newform.gluebound, newform.transparencies, newform.twoholepunched, newform.folded,
newform.proofrequired)
                accountdetails.saveaccountdetails(jobnumber, newform.userid,
newform.staffinitials, department, newform.requestdate, newform.numberofcopies,
totalprice)
                If Login.username.Text = "8000" Then
                    MsgBox("Print job form submitted successfully")
                    Reprographics_usermenu.Show()
                    Me.Close()

                Else
                    MsgBox("Print job form submitted successfully")
                    User_Menu.Show()
                    Me.Close()
                End If
            ElseIf canjobproceed = False Then
                MsgBox("Print job unsuccessful because you have insufficient print
credits")
            ElseIf totalprice = 0 Then
                MsgBox("Nothing has been entered")
            End If
        ElseIf uservalid = False Then
            MsgBox("UserID invalid, please enter a valid UserID")
        ElseIf securitycodeentered.Text <> securitycode.Text Then
            MsgBox("Wrong security code entered, please enter the security code
again")

        End If

    End Sub

    Private Sub Print_job_form_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim randomnumber As New Random
        Dim x As Integer
        x = randomnumber.Next(1, 100000)
        securitycode.AppendText(x)
        userid.Text = Login.username.Text
    End Sub

    Private Sub cancel_Click_1(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cancel.Click
        If Login.username.Text = "8000" Then
            Reprographics_usermenu.Show()
            Me.Close()
        Else
            User_Menu.Show()
            Me.Close()
        End If
```

```
    End Sub
End Class
```

Print job queue




```
Public Class Print_job_queue

    Private Sub update_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles updatequeue.Click
        Dim update As New addtoqueue
        update.addformtoqueue()
    End Sub

    Private Sub viewselectedjob_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles viewselectedjob.Click
        viewprintjobforms.Show()
    End Sub

    Private Sub Print_job_queue_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim update As New addtoqueue
        update.addformtoqueue()
    End Sub

    Private Sub mainmenu_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mainmenu.Click
        Reprographics_usermenu.Show()
        Me.Close()
    End Sub

    Private Sub removejob_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles removejob.Click
        Dim topofstack As String
        Dim changeprintjobformstatus As New printjobform
        Dim topstack As New addtoqueue
        topofstack = topstack.addformtoqueue()
        changeprintjobformstatus.jobcompleted(topofstack)
```

```
        Queue.Items.Remove(topofstack)
    End Sub
End Class
```

Print credits (class)

```
Public Class printcredits
    Function userid(ByVal number As Integer)
        Dim message As New System.Xml.XmlDocument
        message.Load("printcredits.xml")
        Dim printcreditdetails As System.Xml.XmlNodeList =
message.SelectNodes("userprintcreditprofile/printcredit")
        userid =
printcreditdetails.Item(number).SelectSingleNode("userid").InnerText()
    End Function
    Function printcredits(ByVal number As Integer)
        Dim message As New System.Xml.XmlDocument
        message.Load("printcredits.xml")
        Dim printcreditdetails As System.Xml.XmlNodeList =
message.SelectNodes("userprintcreditprofile/printcredit")
        printcredits =
printcreditdetails.Item(number).SelectSingleNode("printcredits").InnerText()
    End Function
    Function Length()
        Dim message As New System.Xml.XmlDocument
        message.Load("printcredits.xml")
        Dim printcreditdetails As System.Xml.XmlNodeList =
message.SelectNodes("userprintcreditprofile/printcredit")
        Return printcreditdetails.Count
    End Function
    Function saveprintcredits(ByVal userid As Integer, ByVal printcredits As Integer)
        Dim printcreditdetailsxml = XDocument.Load("printcredits.xml")
        Dim printcreditdetail As XElement = _
         <printcredit>
             <userid>[redacted] userid [redacted]</userid>
             <printcredits>[redacted] printcredits [redacted]</printcredits>
         </printcredit>

        printcreditdetailsxml.<userprintcreditprofile>(0).Add(printcreditdetail)
        printcreditdetailsxml.Save("printcredits.xml")
        Return True
    End Function
    Function updateprintcredits(ByVal userid As Integer, ByVal value As Integer)
        Dim checkprintcredits As New printcredits
        Dim currentprintcreditnumber As Integer
        Dim updatedprintcreditnumber As Integer
        Dim profilenumber As Integer
        Dim findprofilenumber As New finduser

        profilenumber = findprofilenumber.finduserprofile(userid)
        currentprintcreditnumber = checkprintcredits.printcredits(profilenumber)
        updatedprintcreditnumber = currentprintcreditnumber - value
        If updatedprintcreditnumber <= 0 Then
            updateprintcredits = False
        Else
            updateprintcredits = True
            changeprintcredits(profilenumber, updatedprintcreditnumber)
        End If
        Return updateprintcredits
    End Function
    Sub changeprintcredits(ByVal profilenumber As Integer, ByVal newvalue As Integer)
```

```
        Dim printcreditxml = XDocument.Load("printcredits.xml")

printcreditxml.<userprintcreditprofile>(0).<printcredit>(profilenumber).<printcredits>
(0).SetValue(newvalue)
        printcreditxml.Save("printcredits.xml")
    End Sub

    Function lookupprintcredits(ByVal number As Integer)
        Dim printcreditdetails As New System.Xml.XmlDocument
        printcreditdetails.Load("printcredits.xml")
        Dim printcreditdetail As System.Xml.XmlNodeList =
printcreditdetails.SelectNodes("userprintcreditprofile/printcredit")
        lookupprintcredits =
printcreditdetail.Item(number).SelectSingleNode("printcredits").InnerText()
    End Function
End Class
```

Print job form (class)

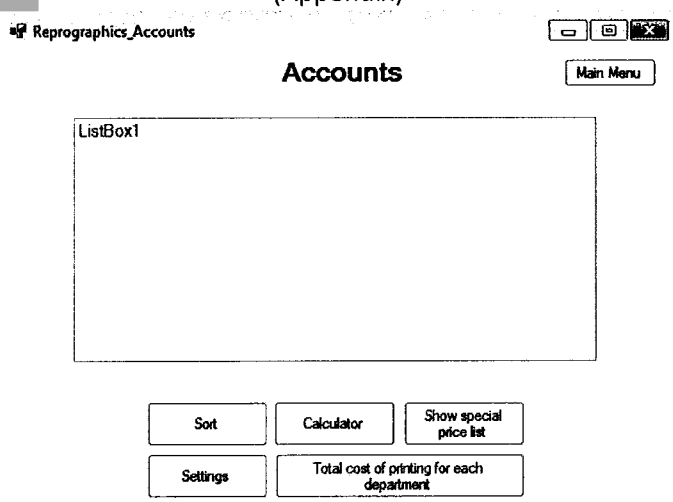
```
Public Class printjobform
    Function jobnumber(ByVal number As Integer)
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        jobnumber =
printjobformitems.Item(number).SelectSingleNode("jobnumber").InnerText()
    End Function
    Function userid(ByVal number As Integer)
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        userid = printjobformitems.Item(number).SelectSingleNode("userid").InnerText()
    End Function

    Function staffinitials(ByVal number As Integer)
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        staffinitials =
printjobformitems.Item(number).SelectSingleNode("staffinitials").InnerText()
    End Function
    Function requestdate(ByVal number As Integer)
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        requestdate =
printjobformitems.Item(number).SelectSingleNode("requestdate").InnerText()
    End Function
    Function copysuppliedas(ByVal number As Integer)
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        copysuppliedas =
printjobformitems.Item(number).SelectSingleNode("copysuppliedas").InnerText()
    End Function
    Function papertype(ByVal number As Integer)
```

```
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        papertype =
printjobformitems.Item(number).SelectSingleNode("papertype").InnerText()
    End Function
    Function papersize(ByVal number As Integer)
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        papersize =
printjobformitems.Item(number).SelectSingleNode("papersize").InnerText()
    End Function
    Function numberofcopies(ByVal number As Integer)
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        numberofcopies =
printjobformitems.Item(number).SelectSingleNode("numberofcopies").InnerText()
    End Function
    Function papercolour(ByVal number As Integer)
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        papercolour =
printjobformitems.Item(number).SelectSingleNode("papercolour").InnerText()
    End Function
    Function backtoback(ByVal number As Integer)
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        backtoback =
printjobformitems.Item(number).SelectSingleNode("backtoback").InnerText()
    End Function
    Function collated(ByVal number As Integer)
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        collated =
printjobformitems.Item(number).SelectSingleNode("collated").InnerText()
    End Function
    Function stapled(ByVal number As Integer)
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        stapled =
printjobformitems.Item(number).SelectSingleNode("stapled").InnerText()
    End Function
    Function wirobinding(ByVal number As Integer)
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        wirobinding =
printjobformitems.Item(number).SelectSingleNode("wirobinding").InnerText()
    End Function
    Function laminate(ByVal number As Integer)
```

```
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        laminate =
printjobformitems.Item(number).SelectSingleNode("laminate").InnerText()
    End Function
    Function gluebound(ByVal number As Integer)
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        gluebound =
printjobformitems.Item(number).SelectSingleNode("gluebound").InnerText()
    End Function
    Function transparencies(ByVal number As Integer)
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        transparencies =
printjobformitems.Item(number).SelectSingleNode("transparencies").InnerText()
    End Function
    Function twoholepunched(ByVal number As Integer)
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        twoholepunched =
printjobformitems.Item(number).SelectSingleNode("twoholepunched").InnerText()
    End Function
    Function folded(ByVal number As Integer)
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        folded = printjobformitems.Item(number).SelectSingleNode("folded").InnerText()
    End Function
    Function proofrequired(ByVal number As Integer)
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        proofrequired =
printjobformitems.Item(number).SelectSingleNode("proofrequired").InnerText()
    End Function
    Function jobcomplete(ByVal number As Integer)
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        jobcomplete =
printjobformitems.Item(number).SelectSingleNode("completed").InnerText()
    End Function
    Function Length()
        Dim printjob As New System.Xml.XmlDocument
        printjob.Load("printjobform.xml")
        Dim printjobformitems As System.Xml.XmlNodeList =
printjob.SelectNodes("jobs/printjobform")
        Return printjobformitems.Count
    End Function
    Function Add(ByVal jobnumber As Integer, ByVal userid As Integer, ByVal
staffinitials As String, ByVal requestdate As String, ByVal copysuppliedas As String,
ByVal printtype As String, ByVal papertype As String, ByVal papersize As String, ByVal
```

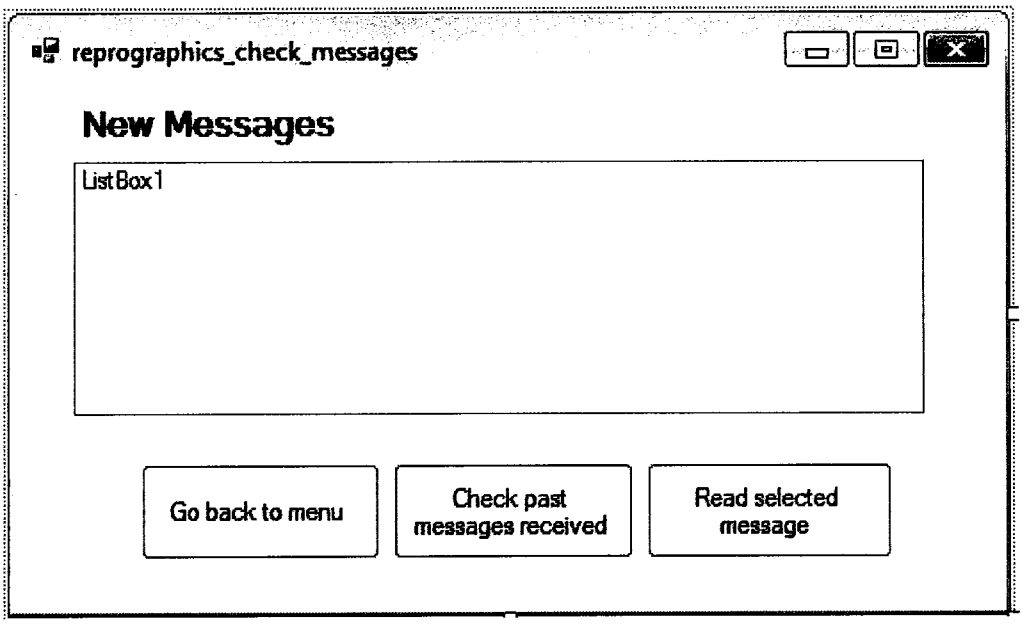
```
numberofcopies As Integer, ByVal papercolour As String, ByVal backtoback As Boolean,
ByVal collated As Boolean, ByVal stapled As Boolean, ByVal wirobinding As Boolean,
ByVal laminate As Boolean, ByVal gluebound As Boolean, ByVal transparencies As
Boolean, ByVal twoholepunched As Boolean, ByVal folded As Boolean, ByVal proofrequired
As Boolean)
        Dim printjobformxml = XDocument.Load("printjobform.xml")
        Dim printjobform As XElement = _
         <printjobform>
             <jobnumber>[redacted] jobnumber [redacted]</jobnumber>
             <userid>[redacted] userid [redacted]</userid>
             <staffinitials>[redacted] staffinitials [redacted]</staffinitials>
             <requestdate>[redacted] requestdate [redacted]</requestdate>
             <copysuppliedas>[redacted] copysuppliedas [redacted]</copysuppliedas>
             <printtype>[redacted] printtype [redacted]</printtype>
             <papertype>[redacted] papertype [redacted]</papertype>
             <papersize>[redacted] papersize [redacted]</papersize>
             <numberofcopies>[redacted] numberofcopies [redacted]</numberofcopies>
             <papercolour>[redacted] papercolour [redacted]</papercolour>
             <backtoback>[redacted] backtoback [redacted]</backtoback>
             <collated>[redacted] collated [redacted]</collated>
             <stapled>[redacted] stapled [redacted]</stapled>
             <wirobinding>[redacted] wirobinding [redacted]</wirobinding>
             <laminate>[redacted] laminate [redacted]</laminate>
             <gluebound>[redacted] gluebound [redacted]</gluebound>
             <transparencies>[redacted] transparencies [redacted]</transparencies>
             <twoholepunched>[redacted] twoholepunched [redacted]</twoholepunched>
             <folded>[redacted] folded [redacted]</folded>
             <proofrequired>[redacted] proofrequired [redacted]</proofrequired>
             <completed>false</completed>
         </printjobform>
        printjobformxml.<jobs>(0).Add(printjobform)
        printjobformxml.Save("printjobform.xml")
        Return True
    End Function
    Sub jobcompleted(ByVal job As String)

        Dim printjobformxml = XDocument.Load("printjobform.xml")
        Dim findjob As New findjob
        Dim jobnumber As Integer
        jobnumber = findjob.findrightjob(job)

printjobformxml.<jobs>(0).<printjobform>(jobnumber).<completed>(0).SetValue("true")
        printjobformxml.Save("printjobform.xml")
    End Sub
End Class
```

Reprographics accounts



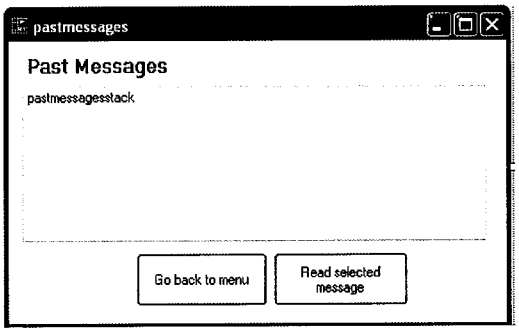

```
Public Class Reprographics_Accounts
    Private Sub calculator_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles calculator.Click
        Form1.Show()
    End Sub

    Private Sub Reprographics_Accounts_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim update As New addtoqueue
        update.addaccountdetails()
    End Sub

    Private Sub settings_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles settings.Click
        accounts_settings.Show()
        Me.Close()
    End Sub

    Private Sub showspecialpricelist_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles showspecialpricelist.Click
        show_special_price_list.Show()
        Me.Close()
    End Sub

    Private Sub sort_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles sort.Click
        accounts_sort.Show()
        Me.Close()
    End Sub

    Private Sub mainmenu_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mainmenu.Click
        Reprographics_usermenu.Show()
        Me.Close()
    End Sub

    Private Sub totalcostofprintingforeachdepartment_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
totalcostofprintingforeachdepartment.Click
        total_cost_of_printing_for_each_department.Show()
        Me.Close()
    End Sub
End Class
```

Reprographics check messages




```
Public Class reprographics_check_messages

    Private Sub gobacktomenu_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles gobacktomenu.Click
        Reprographics_usermenu.Show()
        Me.Close()
    End Sub
    Private Sub reprographics_check_messages_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
        Dim update As New addtoqueue
        update.addnewmessagetostack()
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        Dim messageread As New user_sent_message
        Dim message As String
        message = newmessagestack.SelectedItem
        messageread.setmessagetoread(message)
        view_selected_new_message.Show()
        Me.Close()
    End Sub

    Private Sub pastmessages_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles checkpastmessages.Click
        Reprographics_past_messages.Show()
        Me.Close()
    End Sub
End Class
```

Reprographics past messages




```
Public Class Reprographics_past_messages

    Private Sub pastmessages_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim update As New addtoqueue
        update.addpastmessagetoqueue()
    End Sub

    Private Sub mainmenu_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mainmenu.Click
        Reprographics_usermenu.Show()
        Me.Close()
    End Sub

    Private Sub readselectedmessage_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles readselectedmessage.Click
        Dim messageread As New user_sent_message
        view_selected_past_message.Show()
        messageread.setmessagetoread(pastmessagesstack.SelectedValue)
        Me.Close()
    End Sub
End Class
```

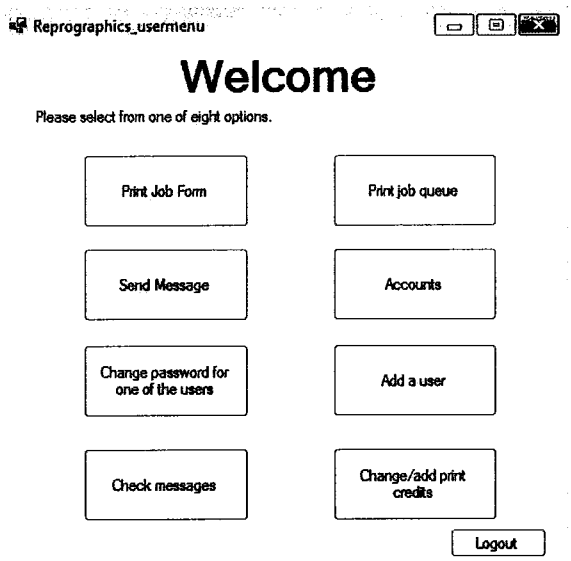
Reprographics sent messages (class)

```
Public Class reprographics_sent_messages
    Function messagenumber(ByVal number As Integer)
        Dim message As New System.Xml.XmlDocument
        message.Load("reprographicssentmessages.xml")
        Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("reprographicssentmessages/message")
        messagenumber =
messagedetails.Item(number).SelectSingleNode("messagenumber").InnerText()
    End Function
    Function userid(ByVal number As Integer)
        Dim message As New System.Xml.XmlDocument
        message.Load("reprographicssentmessages.xml")
        Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("reprographicssentmessages/message")
        userid = messagedetails.Item(number).SelectSingleNode("userid").InnerText()
    End Function
    Function name(ByVal number As Integer)
        Dim message As New System.Xml.XmlDocument
        message.Load("reprographicssentmessages.xml")
        Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("reprographicssentmessages/message")
        name = messagedetails.Item(number).SelectSingleNode("name").InnerText()
    End Function
    Function datesubmitted(ByVal number As Integer)
        Dim message As New System.Xml.XmlDocument
```

```
        message.Load("reprographicssentmessages.xml")
        Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("reprographicssentmessages/message")
        datesubmitted =
messagedetails.Item(number).SelectSingleNode("datesent").InnerText()
    End Function
    Function topic(ByVal number As Integer)
        Dim message As New System.Xml.XmlDocument
        message.Load("reprographicssentmessages.xml")
        Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("reprographicssentmessages/message")
        topic = messagedetails.Item(number).SelectSingleNode("topic").InnerText()
    End Function
    Function submittedmessage(ByVal number As Integer)
        Dim message As New System.Xml.XmlDocument
        message.Load("reprographicssentmessages.xml")
        Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("reprographicssentmessages/message")
        submittedmessage =
messagedetails.Item(number).SelectSingleNode("message").InnerText()
    End Function
    Function read(ByVal number As Integer)
        Dim message As New System.Xml.XmlDocument
        message.Load("reprographicssentmessages.xml")
        Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("reprographicssentmessages/message")
        read = messagedetails.Item(number).SelectSingleNode("messageread").InnerText()
    End Function
    Sub setmessagetoread(ByVal message As String)
        Dim messagexml = XDocument.Load("reprographicssentmessages.xml")
        Dim findmessageprofile As New findjob
        Dim messagenumber As Integer
        messagenumber = findmessageprofile.findrightreprographicsmessage(message)

messagexml.<reprographicssentmessages>(0).<message>(messagenumber).<messageread>(0).Se
tValue("true")
        messagexml.Save("reprographicssentmessages.xml")
    End Sub
    Function Length()
        Dim message As New System.Xml.XmlDocument
        message.Load("reprographicssentmessages.xml")
        Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("reprographicssentmessages/message")
        Return messagedetails.Count()
    End Function
    Function saveregraphicsentmessage(ByVal messagenumber As Integer, ByVal userid As
Integer, ByVal name As String, ByVal datesent As String, ByVal topic As String, ByVal
message As String)
        Dim reprographicsmessagesentdetailsxml =
XDocument.Load("reprographicssentmessages.xml")
        Dim messagesentdetail As XElement = _
         <message>
             <messagenumber>[redacted] messagenumber [redacted]</messagenumber>
             <userid>[redacted] userid [redacted]</userid>
             <name>[redacted] name [redacted]</name>
             <datesent>[redacted] datesent [redacted]</datesent>
             <topic>[redacted] topic [redacted]</topic>
             <message>[redacted] message [redacted]</message>
             <messageread>[redacted] False [redacted]</messageread>
         </message>

reprographicsmessagesentdetailsxml.<reprographicssentmessages>(0).Add(messagesentdetai
l)
        reprographicsmessagesentdetailsxml.Save("reprographicssentmessages.xml")
```

```
        Return True
    End Function
End Class
```

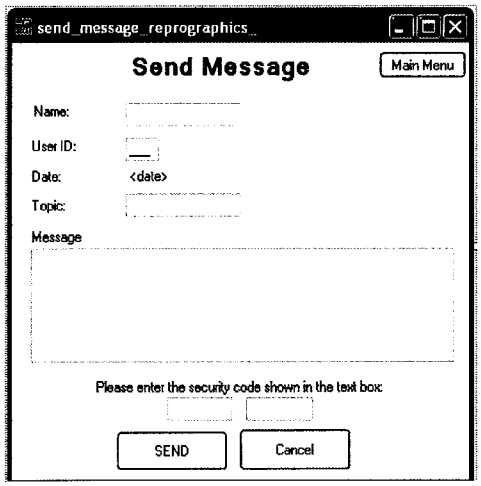
Reprographics user menu




```
Public Class Reprographics_usermenu

    Private Sub printjobform_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles printjobform.Click
        Print_job_form.Show()
        Me.Close()
    End Sub

    Private Sub printjobqueue_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles printjobqueue.Click
        Print_job_queue.Show()
        Me.Close()
    End Sub

    Private Sub accounts_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles accounts.Click
        Reprographics_Accounts.Show()
        Me.Close()
    End Sub

    Private Sub addauser_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles addauser.Click
        Add_a_user.Show()
        Me.Close()
    End Sub

    Private Sub changepasswordforoneoftheusers_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles changepasswordforoneoftheusers.Click
        change_username_password_of_user.Show()
        Me.Close()
    End Sub

    Private Sub checkmessages_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles checkmessages.Click
```

```
        reprographics_check_messages.Show()
        Me.Close()
    End Sub

    Private Sub changeaddprintcredits_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles changeaddprintcredits.Click
        change_add_print_credits.Show()
        Me.Close()
    End Sub

    Private Sub sendamessage_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles sendamessage.Click
        send_message__reprographics_.Show()
        Me.Close()
    End Sub

    Private Sub logout_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles logout.Click
        Login.username.Clear()
        Login.password.Clear()
        Login.Show()
        Me.Close()
    End Sub

    Private Sub Reprographics_usermenu_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

    End Sub
End Class
```

Send message (reprographics)

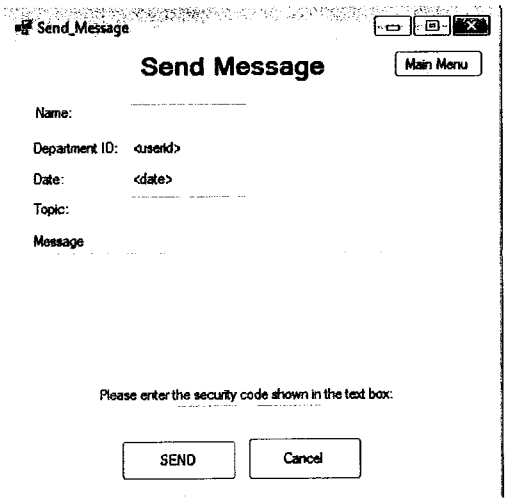



```
ublic Class send_message__reprographics_

    Private Sub send_message__reprographics__Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
        Dim randomnumber As New Random
        Dim x As Integer
        x = randomnumber.Next(1, 1000000)
        securitycode.AppendText(x)
        todaysdate.Text = Now.Date
    End Sub

    Private Sub send_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles send.Click
```

```
        Dim check As New reprographics_sent_messages
        Dim getmessagedetails As New getmessagedetails
        Dim savedetails As New reprographics_sent_messages
        Dim messagenumber As Integer = check.Length
        Dim sendthismessage As Boolean = True
        If securitycodeentered.Text = securitycode.Text Then
            messagenumber = messagenumber + 1

            If nameentered.Text <> "" Then
                getmessagedetails.getname = nameentered.Text
                sendthismessage = True
            Else
                MsgBox("You have not entered your name")
                sendthismessage = False
            End If
            If userid.Text <> "" Then
                getmessagedetails.getuserid = userid.Text
                sendthismessage = True
            Else
                MsgBox("You have not entered the userID")
                sendthismessage = False
            End If
            If topicentered.Text <> "" Then
                getmessagedetails.gettopic = topicentered.Text
                sendthismessage = True
            Else
                MsgBox("You have not entered the topic")
                sendthismessage = False
            End If
            If message.Text <> "" Then
                getmessagedetails.getmessage = message.Text
                sendthismessage = True
            Else
                MsgBox("You have not entered the message")
                sendthismessage = False
            End If
            getmessagedetails.getdatesubmitted = todaysdate.Text
            getmessagedetails.getuserid = userid.Text
            If sendthismessage = True Then
                savedetails.saveregraphicsentmessage(messagenumber,
getmessagedetails.getuserid, getmessagedetails.getname,
getmessagedetails.getdatesubmitted, getmessagedetails.gettopic,
getmessagedetails.getmessage)
                MsgBox("Message sent to reprographics successfully")
                User_Menu.Show()
                Me.Close()
            End If
        Else
            MsgBox("Wrong or no security code entered, please enter the security code
again")
        End If
    End Sub

    Private Sub mainmenu_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mainmenu.Click
        Reprographics_usermenu.Show()
        Me.Close()
    End Sub

    Private Sub Cancel_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Cancel.Click
        Reprographics_usermenu.Show()
        Me.Close()
    End Sub
```

```
End Class
```

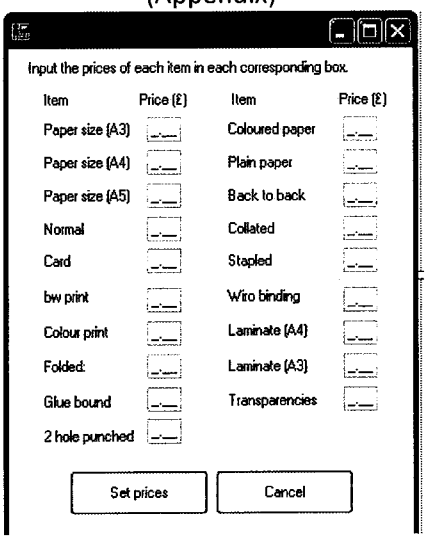
Send message




```
Public Class Send_Message

    Private Sub Mainmenu_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mainmenu.Click
        User_Menu.Show()
        Me.Close()
    End Sub

    Private Sub cancel_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Cancel.Click
        User_Menu.Show()
        Me.Close()
    End Sub

    Private Sub Send_Message_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim randomnumber As New Random
        Dim x As Integer
        x = randomnumber.Next(1, 1000000)
        securitycode.AppendText(x)
        userid.Text = Login.username.Text
        todaysdate.Text = Now.Date
    End Sub

    Private Sub send_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles send.Click
        Dim messageclass As New user_sent_message
        Dim getmessagedetails As New getmessagedetails
        Dim findmessagenumber As Integer = messageclass.length
        Dim sendthismessage As Boolean = False
        If securitycodeentered.Text = securitycode.Text Then
            findmessagenumber = findmessagenumber + 1
            If nameentered.Text <> "" Then
                getmessagedetails.getname = nameentered.Text
                sendthismessage = True
            Else
                MsgBox("You have not entered your name")
                sendthismessage = False
            End If
            If topicentered.Text <> "" Then
```

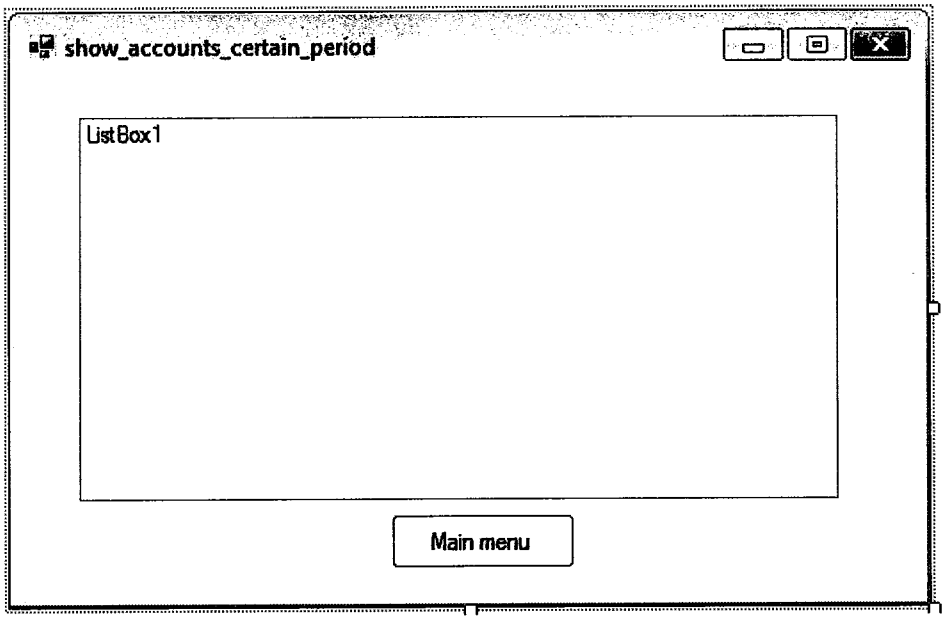
```
                getmessagedetails.gettopic = topicentered.Text
                sendthismessage = True
            Else
                MsgBox("You have not entered the topic")
                sendthismessage = False
            End If
            If message.Text <> "" Then
                getmessagedetails.getmessage = message.Text
                sendthismessage = True
            Else
                MsgBox("You have not entered the message")
                sendthismessage = False
            End If
            getmessagedetails.getdatesubmitted = todaysdate.Text
            getmessagedetails.getuserid = userid.Text
            If sendthismessage = True Then
                messageclass.addmessage(findmessagenumber,
getmessagedetails.getuserid, getmessagedetails.getname,
getmessagedetails.getdatesubmitted, getmessagedetails.gettopic,
getmessagedetails.getmessage)
                MsgBox("Message sent to reprographics successfully")
                User_Menu.Show()
                Me.Close()
            End If
        Else
            MsgBox("Wrong or no security code entered, please enter the security code
again")
        End If
    End Sub
End Class
```

Set new prices on everything




```
Public Class Set_new_prices_on_everything

    Private Sub Cancel_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Cancel.Click
        Reprographics_Accounts.Show()
        Me.Close()
    End Sub

    Private Sub setprices_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles setprices.Click
        Dim price As New getnewpricesoneverything
        Dim pricelist As New special_price_list

        price.getpapersizea3 = papersizea3.Text
        price.getpapersizea4 = papersizea4.Text
        price.getpapersizea5 = papersizea5.Text
        price.getnormal = normal.Text
        price.getcard = card.Text
        price.getbwprint = bwprint.Text
        price.getcolouredprint = colourprint.Text
        price.getcolourpaper = colouredpaper.Text
        price.getplainpaper = plainpaper.Text
        price.getbacktoback = backtoback.Text
        price.getcollated = collated.Text
        price.getstapled = stapled.Text
        price.getwirobinding = wirobinding.Text
        price.getlaminatea4 = laminatea4.Text
        price.getlaminatea3 = laminatea3.Text
        price.getgluebound = gluebound.Text
        price.gettransparencies = transparencies.Text
        price.gettwoholepunched = twoholepunched.Text
        price.getfolded = folded.Text

        pricelist.Add(price.getpapersizea3, price.getpapersizea4,
price.getpapersizea5, price.getnormal, price.getcard, price.getbwprint,
price.getcolouredprint, price.getcolourpaper, price.getplainpaper,
price.getbacktoback, price.getcollated, price.getstapled, price.getwirobinding,
price.getlaminatea4, price.getlaminatea3, price.getgluebound, price.gettransparencies,
price.gettwoholepunched, price.getfolded)
        MsgBox("All the item prices have been successfully set")
        Reprographics_Accounts.Show()
        Me.Close()
    End Sub
End Class
```

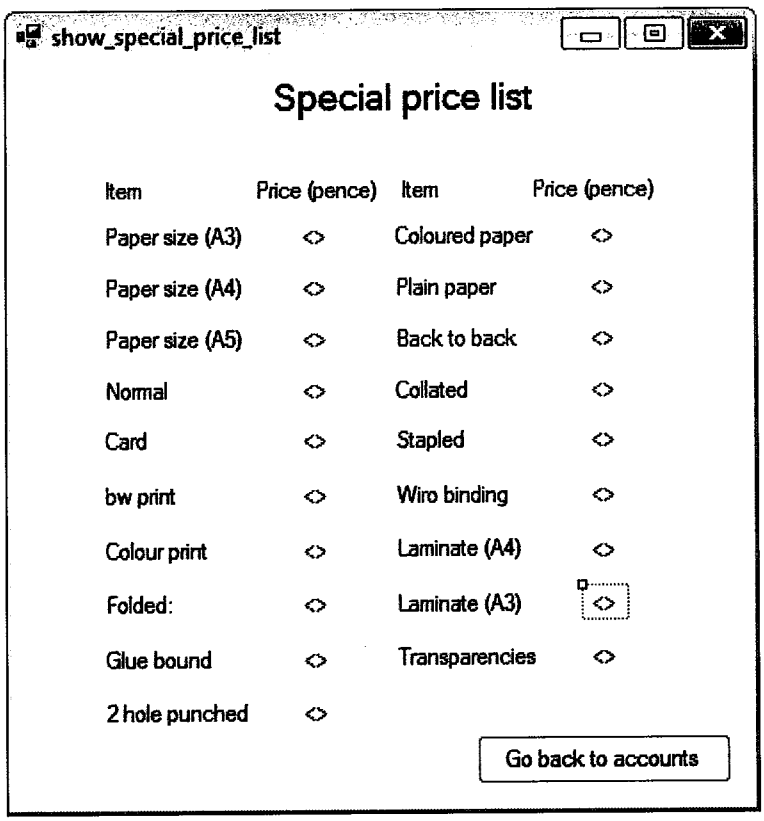
Show accounts certain period




```
Public Class show_accounts_certain_period

    Private Sub show_accounts_certain_period_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load

        Dim length As Integer
        Dim number As Integer = 0
        Dim accounts As New accounts
        Dim type As String
        type = accounts_sort.sortoptions.Text
        length = accounts.Length
        Dim completed As Boolean = True
        Dim temporystoredvalue As Integer
        Dim array(length) As Integer
        Dim arrays(length) As Integer
        Dim sortlist(length) As Integer
        Dim usernumber As Integer
        Dim find As New findjob
        Dim datesubmitted(length) As String
        Dim numbers As Integer = 1
        Dim nomoreswaps As Boolean = False
        For value = 1 To length
            datesubmitted(value) = accounts.datesubmitted(number)
            number = number + 1
        Next
        For value = 1 To length
            If SortableDate(datesubmitted(value)) >=
SortableDate(account_sort_certain_period.datefrom.Text) And
SortableDate(datesubmitted(value)) <=
SortableDate(account_sort_certain_period.dateto.Text) Then
                usernumber =
find.findrightaccountdatesubmitteddetails(datesubmitted(value))
                arrays(numbers) = usernumber
                numbers = numbers + 1
                ListBox1.Items.Add(accounts.userid(usernumber) & "        " &
accounts.staffinitials(usernumber) & "                " &
```

```
accounts.numberofcopies(usernumber) & "                " &
accounts.datesubmitted(usernumber) & "            £" &
accounts.totalprice(usernumber))
            End If
        Next
        If account_sort_certain_period.sortoptions.Text = "Date submitted" Then
            For value = 1 To length
                datesubmitted(value) = accounts.datesubmitted(array(value))
            Next
            For value = 1 To length

                usernumber =
find.findrightaccountdatesubmitteddetails(datesubmitted(value))
                ListBox1.Items.Add(accounts.userid(usernumber) & "        " &
accounts.staffinitials(usernumber) & "                " &
accounts.numberofcopies(usernumber) & "                " &
accounts.datesubmitted(usernumber) & "            £" &
accounts.totalprice(usernumber))
            Next
            completed = True
        ElseIf account_sort_certain_period.sortoptions.Text = "Number of copies" Then
            number = 0
            For value = 1 To length
                array(value) = accounts.numberofcopies(arrays(number))
                number = number + 1
            Next
            Do
                nomoreswaps = True
                For value = 1 To length
                    If array(value) > array(value + 1) Then
                        nomoreswaps = False
                        temporystoredvalue = array(value)
                        array(value) = array(value + 1)
                        array(value + 1) = temporystoredvalue
                    End If
                Next
            Loop Until nomoreswaps = True
            For value = 1 To length
                'usernumber = find.findrightaccountnumberofcopiesdetails(array(value))
                ListBox1.Items.Add(accounts.userid(array(value)) & "        " &
accounts.staffinitials(array(value)) & "                " &
accounts.numberofcopies(array(value)) & "                " &
accounts.datesubmitted(array(value)) & "            £" &
accounts.totalprice(array(value)))
            Next
        End If
    End Sub
    Function SortableDate(ByVal Datevalue As String)
        Dim reorderdate As String
        reorderdate = Datevalue.Substring(8, 2) & Datevalue.Substring(3, 2) &
Datevalue.Substring(0, 2)
        SortableDate = reorderdate
        Return SortableDate
    End Function

    Private Sub reprographicsmenu_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles reprographicsmenu.Click
        Reprographics_Accounts.Show()
        Me.Close()
    End Sub
End Class
```

Show special price list




```
Public Class show_special_price_list

    Private Sub goback_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles goback.Click
        Reprographics_Accounts.Show()
        Me.Close()
    End Sub

    Private Sub show_special_price_list_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim lookupprice As New special_price_list
        papersizea3.Text = lookupprice.papersizea3
        papersizea4.Text = lookupprice.papersizea4
        papersizea5.Text = lookupprice.papersizea5
        normal.Text = lookupprice.normal
        card.Text = lookupprice.card
        bwprint.Text = lookupprice.bwprint
        colourprint.Text = lookupprice.colourprint
        folded.Text = lookupprice.folded
        gluebound.Text = lookupprice.gluebound
        twoholepunched.Text = lookupprice.twoholepunched
        colouredpaper.Text = lookupprice.papercolour
        plainpaper.Text = lookupprice.plainpaper
        backtoback.Text = lookupprice.backtoback
        collated.Text = lookupprice.collated
        stapled.Text = lookupprice.stapled
        wirobinding.Text = lookupprice.wirobinding
        laminatea4.Text = lookupprice.laminatea4
        laminatea3.Text = lookupprice.laminatea3
        transparencies.Text = lookupprice.transparencies
    End Sub
```

```
End Class
```

Special price list (class)

```
Public Class special_price_list
    Function papersizea3()
        Dim specialpriceitem As New System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
        Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        papersizea3 = speicalpricedetails.Item(0).SelectSingleNode("a3").InnerText()
    End Function
    Function papersizea4()
        Dim specialpriceitem As New System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
        Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        papersizea4 = speicalpricedetails.Item(0).SelectSingleNode("a4").InnerText()
    End Function
    Function papersizea5()
        Dim specialpriceitem As New System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
        Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        papersizea5 = speicalpricedetails.Item(0).SelectSingleNode("a5").InnerText()
    End Function
    Function normal()
        Dim specialpriceitem As New System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
        Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        normal = speicalpricedetails.Item(0).SelectSingleNode("normal").InnerText()
    End Function
    Function card()
        Dim specialpriceitem As New System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
        Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        card = speicalpricedetails.Item(0).SelectSingleNode("card").InnerText()
    End Function
    Function papercolour()
        Dim specialpriceitem As New System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
        Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        papercolour =
speicalpricedetails.Item(0).SelectSingleNode("papercolour").InnerText()
    End Function
    Function bwprint()
        Dim specialpriceitem As New System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
        Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        bwprint =
speicalpricedetails.Item(0).SelectSingleNode("blackandwhite").InnerText()
    End Function
    Function colourprint()
        Dim specialpriceitem As New System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
        Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        colourprint =
speicalpricedetails.Item(0).SelectSingleNode("colour").InnerText()
    End Function
```

```
    Function plainpaper()
        Dim specialpriceitem As New System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
        Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        plainpaper =
speicalpricedetails.Item(0).SelectSingleNode("blackandwhite").InnerText()
    End Function
    Function backtoback()
        Dim specialpriceitem As New System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
        Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        backtoback =
speicalpricedetails.Item(0).SelectSingleNode("backtoback").InnerText()
    End Function
    Function collated()
        Dim specialpriceitem As New System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
        Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        collated =
speicalpricedetails.Item(0).SelectSingleNode("collated").InnerText()
    End Function
    Function stapled()
        Dim specialpriceitem As New System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
        Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        stapled = speicalpricedetails.Item(0).SelectSingleNode("stapled").InnerText()
    End Function
    Function wirobinding()
        Dim specialpriceitem As New System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
        Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        wirobinding =
speicalpricedetails.Item(0).SelectSingleNode("wirobinding").InnerText()
    End Function
    Function laminatea4()
        Dim specialpriceitem As New System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
        Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        laminatea4 =
speicalpricedetails.Item(0).SelectSingleNode("laminatea4").InnerText()
    End Function
    Function laminatea3()
        Dim specialpriceitem As New System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
        Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        laminatea3 =
speicalpricedetails.Item(0).SelectSingleNode("laminatea3").InnerText()
    End Function
    Function gluebound()
        Dim specialpriceitem As New System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
        Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        gluebound =
speicalpricedetails.Item(0).SelectSingleNode("gluebound").InnerText()
    End Function
    Function transparencies()
```

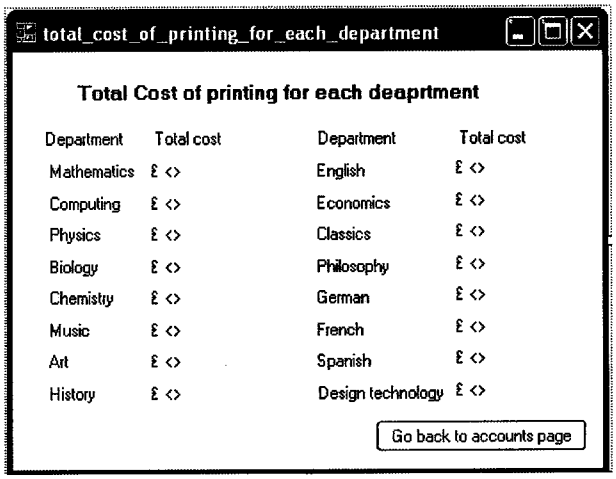
```
        Dim specialpriceitem As New System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
        Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        transparencies =
speicalpricedetails.Item(0).SelectSingleNode("transparencies").InnerText()
    End Function
    Function twoholepunched()
        Dim specialpriceitem As New System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
        Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        twoholepunched =
speicalpricedetails.Item(0).SelectSingleNode("twoholepunched").InnerText()
    End Function
    Function folded()
        Dim specialpriceitem As New System.Xml.XmlDocument
        specialpriceitem.Load("specialprices.xml")
        Dim speicalpricedetails As System.Xml.XmlNodeList =
specialpriceitem.SelectNodes("prices/specialprice")
        folded = speicalpricedetails.Item(0).SelectSingleNode("folded").InnerText()
    End Function
    Function Add(ByVal a3 As String, ByVal a4 As String, ByVal a5 As String, ByVal
normal As String, ByVal card As String, ByVal bwprint As String, ByVal colourprint As
String, ByVal papercolour As String, ByVal plainpaper As String, ByVal backtoback As
String, ByVal collated As String, ByVal stapled As String, ByVal wirobinding As
String, ByVal laminatea4 As String, ByVal laminatea3 As String, ByVal gluebound As
String, ByVal transparencies As String, ByVal twoholepunched As String, ByVal folded
As String)
        Dim specialpricexml = XDocument.Load("specialprices.xml")
        specialpricexml.<prices>(0).RemoveNodes()
        Dim specialprice As XElement = _
         <specialprice>
             <a3>[redacted] a3 [redacted]</a3>
             <a4>[redacted] a4 [redacted]</a4>
             <a5>[redacted] a5 [redacted]</a5>
             <normal>[redacted] normal [redacted]</normal>
             <card>[redacted] card [redacted]</card>
             <blackandwhite>[redacted] bwprint [redacted]</blackandwhite>
             <colour>[redacted] colourprint [redacted]</colour>
             <papercolour>[redacted] papercolour [redacted]</papercolour>
             <paperbandw>[redacted] plainpaper [redacted]</paperbandw>
             <backtoback>[redacted] backtoback [redacted]</backtoback>
             <collated>[redacted] collated [redacted]</collated>
             <stapled>[redacted] stapled [redacted]</stapled>
             <wirobinding>[redacted] wirobinding [redacted]</wirobinding>
             <laminatea4>[redacted] laminatea4 [redacted]</laminatea4>
             <laminatea3>[redacted] laminatea3 [redacted]</laminatea3>
             <gluebound>[redacted] gluebound [redacted]</gluebound>
             <transparencies>[redacted] transparencies [redacted]</transparencies>
             <twoholepunched>[redacted] twoholepunched [redacted]</twoholepunched>
             <folded>[redacted] folded [redacted]</folded>
         </specialprice>
        specialpricexml.<prices>(0).Add(specialprice)
        specialpricexml.Save("specialprices.xml")
        Return True
    End Function
    Sub changespecialprice(ByVal item As String, ByVal newprice As String)
        If item = "a3" Then
            a3(item, newprice)
        ElseIf item = "a4" Then
            a4(item, newprice)
        ElseIf item = "a5" Then
            a5(item, newprice)
```

```
        ElseIf item = "normal" Then
            normal(item, newprice)
        ElseIf item = "card" Then
            card(item, newprice)
        ElseIf item = "blackandwhite" Then
            blackandwhite(item, newprice)
        ElseIf item = "colour" Then
            colour(item, newprice)
        ElseIf item = "papercolour" Then
            papercolour(item, newprice)
        ElseIf item = "paperbandw" Then
            paperbandw(item, newprice)
        ElseIf item = "backtoback" Then
            backtoback(item, newprice)
        ElseIf item = "collated" Then
            collated(item, newprice)
        ElseIf item = "stapled" Then
            stapled(item, newprice)
        ElseIf item = "wirobinding" Then
            wirobinding(item, newprice)
        ElseIf item = "laminatea4" Then
            laminatea4(item, newprice)
        ElseIf item = "laminatea3" Then
            laminatea3(item, newprice)
        ElseIf item = "gluebound" Then
            gluebound(item, newprice)
        ElseIf item = "transparencies" Then
            transparencies(item, newprice)
        ElseIf item = "twoholepunched" Then
            twoholepunched(item, newprice)
        ElseIf item = "folded" Then
            folded(item, newprice)
        End If
    End Sub
    Sub a3(ByVal item As String, ByVal newprice As String)
        Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<a3>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
    End Sub
    Sub a4(ByVal item As String, ByVal newprice As String)
        Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<a4>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
    End Sub
    Sub a5(ByVal item As String, ByVal newprice As String)
        Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<a5>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
    End Sub
    Sub normal(ByVal item As String, ByVal newprice As String)
        Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<normal>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
    End Sub
    Sub card(ByVal item As String, ByVal newprice As String)
        Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<card>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
    End Sub
    Sub blackandwhite(ByVal item As String, ByVal newprice As String)
        Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<blackandwhite>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
    End Sub
```

```
    Sub colour(ByVal item As String, ByVal newprice As String)
        Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<colour>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
    End Sub
    Sub papercolour(ByVal item As String, ByVal newprice As String)
        Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<papercolour>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
    End Sub
    Sub paperbandw(ByVal item As String, ByVal newprice As String)
        Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<paperbandw>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
    End Sub
    Sub backtoback(ByVal item As String, ByVal newprice As String)
        Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<backtoback>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
    End Sub
    Sub collated(ByVal item As String, ByVal newprice As String)
        Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<collated>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
    End Sub
    Sub stapled(ByVal item As String, ByVal newprice As String)
        Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<stapled>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
    End Sub
    Sub wirobinding(ByVal item As String, ByVal newprice As String)
        Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<wirobinding>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
    End Sub
    Sub laminatea4(ByVal item As String, ByVal newprice As String)
        Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<laminatea4>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
    End Sub
    Sub laminatea3(ByVal item As String, ByVal newprice As String)
        Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<laminatea3>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
    End Sub
    Sub gluebound(ByVal item As String, ByVal newprice As String)
        Dim specialprices = XDocument.Load("specialprices.xml")
        specialprices.<prices>(0).<specialprice>.<gluebound>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
    End Sub
    Sub transparencies(ByVal item As String, ByVal newprice As String)
        Dim specialprices = XDocument.Load("specialprices.xml")

specialprices.<prices>(0).<specialprice>.<transparencies>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
    End Sub
    Sub twoholepunched(ByVal item As String, ByVal newprice As String)
        Dim specialprices = XDocument.Load("specialprices.xml")

specialprices.<prices>(0).<specialprice>.<twoholepunched>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
    End Sub
    Sub folded(ByVal item As String, ByVal newprice As String)
        Dim specialprices = XDocument.Load("specialprices.xml")
```

```
        specialprices.<prices>(0).<specialprice>.<folded>(0).SetValue(newprice)
        specialprices.Save("specialprices.xml")
    End Sub
End Class
```

Total cost of printing for each department



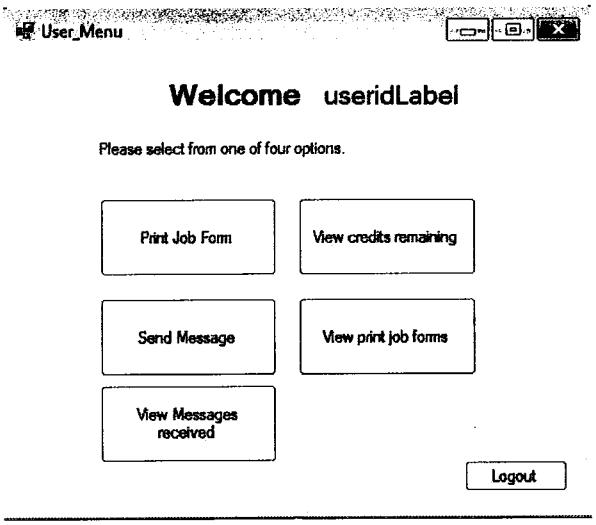

```
Public Class total_cost_of_printing_for_each_department
    Dim number = 0
    Dim getaccountdetails As New accounts
    Dim totalcost As Integer = 0
    Private Sub total_cost_of_printing_for_each_department_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Dim length As Integer
        length = getaccountdetails.Length
        Do
            If getaccountdetails.department(number) = "Mathematics" Then
                totalcost = totalcost + getaccountdetails.totalprice(number)
                number = number + 1
            Else
                number = number + 1
            End If
        Loop Until number = length - 1
        Mathematics.Text = totalcost
        number = 0
        totalcost = 0
        Do
            If getaccountdetails.department(number) = "Computing" Then
                totalcost = totalcost + getaccountdetails.totalprice(number)
                number = number + 1
            Else
                number = number + 1
            End If
        Loop Until number = length - 1
        Computing.Text = totalcost
        number = 0
        totalcost = 0
        Do
            If getaccountdetails.department(number) = "Physics" Then
                totalcost = totalcost + getaccountdetails.totalprice(number)
                number = number + 1
            Else
                number = number + 1
            End If
```

```
Loop Until number = length - 1
Physics.Text = totalcost
number = 0
totalcost = 0
Do
    If getaccountdetails.department(number) = "Biology" Then
        totalcost = totalcost + getaccountdetails.totalprice(number)
        number = number + 1
    Else
        number = number + 1
    End If
Loop Until number = length - 1
Biology.Text = totalcost
number = 0
totalcost = 0
Do
    If getaccountdetails.department(number) = "Chemistry" Then
        totalcost = totalcost + getaccountdetails.totalprice(number)
        number = number + 1
    Else
        number = number + 1
    End If
Loop Until number = length - 1
Chemistry.Text = totalcost
number = 0
totalcost = 0
Do
    If getaccountdetails.department(number) = "Music" Then
        totalcost = totalcost + getaccountdetails.totalprice(number)
        number = number + 1
    Else
        number = number + 1
    End If
Loop Until number = length - 1
Music.Text = totalcost
number = 0
totalcost = 0
Do
    If getaccountdetails.department(number) = "Art" Then
        totalcost = totalcost + getaccountdetails.totalprice(number)
        number = number + 1
    Else
        number = number + 1
    End If
Loop Until number = length - 1
Art.Text = totalcost
number = 0
totalcost = 0
Do
    If getaccountdetails.department(number) = "History" Then
        totalcost = totalcost + getaccountdetails.totalprice(number)
        number = number + 1
    Else
        number = number + 1
    End If
Loop Until number = length - 1
History.Text = totalcost
number = 0
totalcost = 0
Do
    If getaccountdetails.department(number) = "English" Then
        totalcost = totalcost + getaccountdetails.totalprice(number)
        number = number + 1
    Else
```

```
        number = number + 1
    End If
Loop Until number = length - 1
English.Text = totalcost
number = 0
totalcost = 0
Do
    If getaccountdetails.department(number) = "Economics" Then
        totalcost = totalcost + getaccountdetails.totalprice(number)
        number = number + 1
    Else
        number = number + 1
    End If
Loop Until number = length - 1
Economics.Text = totalcost
number = 0
totalcost = 0
Do
    If getaccountdetails.department(number) = "Classics" Then
        totalcost = totalcost + getaccountdetails.totalprice(number)
        number = number + 1
    Else
        number = number + 1
    End If
Loop Until number = length - 1
Classics.Text = totalcost
number = 0
totalcost = 0
Do
    If getaccountdetails.department(number) = "Philosophy" Then
        totalcost = totalcost + getaccountdetails.totalprice(number)
        number = number + 1
    Else
        number = number + 1
    End If
Loop Until number = length - 1
Philosophy.Text = totalcost
number = 0
totalcost = 0
Do
    If getaccountdetails.department(number) = "French" Then
        totalcost = totalcost + getaccountdetails.totalprice(number)
        number = number + 1
    Else
        number = number + 1
    End If
Loop Until number = length - 1
french.Text = totalcost
number = 0
totalcost = 0
Do
    If getaccountdetails.department(number) = "German" Then
        totalcost = totalcost + getaccountdetails.totalprice(number)
        number = number + 1
    Else
        number = number + 1
    End If
Loop Until number = length - 1
german.Text = totalcost
number = 0
totalcost = 0
Do
    If getaccountdetails.department(number) = "Spanish" Then
        totalcost = totalcost + getaccountdetails.totalprice(number)
```

```
                number = number + 1
            Else
                number = number + 1
            End If
        Loop Until number = length - 1
        spanish.Text = totalcost
        number = 0
        totalcost = 0
        Do
            If getaccountdetails.department(number) = "Design technology" Then
                totalcost = totalcost + getaccountdetails.totalprice(number)
                number = number + 1
            Else
                number = number + 1
            End If
        Loop Until number = length - 1
        designtechnology.Text = totalcost
        number = 0
        totalcost = 0
    End Sub

    Private Sub back_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles back.Click
        Reprographics_Accounts.Show()
        Me.Close()
    End Sub
End Class
```

User menu




```
Public Class User_Menu

    Private Sub sendmessage_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles sendmessage.Click
        Send_Message.Show()
        Me.Close()
    End Sub
    Private Sub printjobform_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles printjobform.Click
        Print_job_form.Show()
        Me.Close()
    End Sub
```

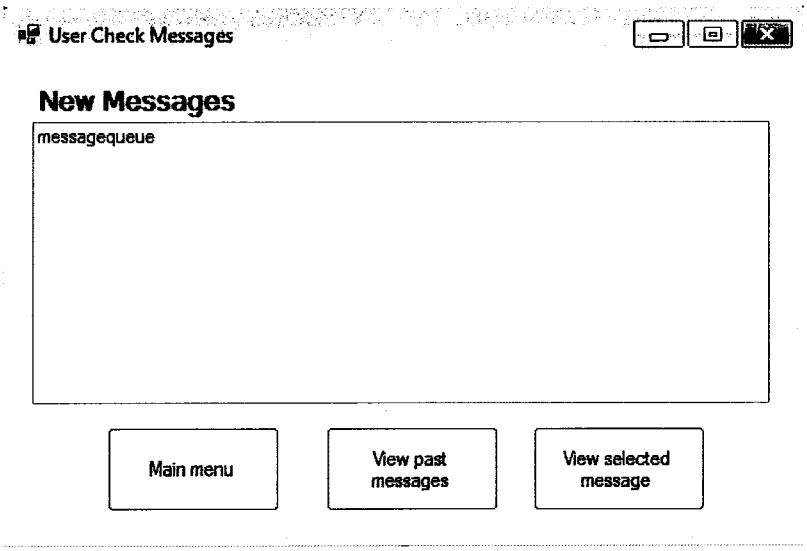
```
    Private Sub creditremaining_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles viewcreditsremaining.Click
        Credit_remaining.Show()
        Me.Close()
    End Sub

    Private Sub viewprintforms_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles viewprintjobforms.Click
        View_print_forms.Show()
        Me.Close()
    End Sub
    Private Sub logout_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles logout.Click
        Login.username.Clear()
        Login.password.Clear()
        Login.Show()
        Me.Close()
    End Sub

    Private Sub User_Menu_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim finddepartmentname As New finduser
        Dim getinformation As New userprofile
        Dim userid As Integer
        userid = finddepartmentname.finduserprofile(Login.username.Text)
        Label.Text = getinformation.department(userid)
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        usercheckmessages.Show()
        Me.Close()
    End Sub
End Class
```

User sent message

```
Public Class user_sent_message
    Function addmessage(ByVal messagenumber As Integer, ByVal userid As Integer, ByVal
name As String, ByVal datesubmitted As String, ByVal topic As String, ByVal message As
String)
        Dim messagedetailsxml = XDocument.Load("messages.xml")
        Dim messagedetail As XElement = _
         <message>
             <messagenumber>[redacted] messagenumber [redacted]</messagenumber>
             <userid>[redacted] userid [redacted]</userid>
             <name>[redacted] name [redacted]</name>
             <datesubmitted>[redacted] datesubmitted [redacted]</datesubmitted>
             <topic>[redacted] topic [redacted]</topic>
             <message>[redacted] message [redacted]</message>
             <read>[redacted] False [redacted]</read>
         </message>
        messagedetailsxml.<messages>(0).Add(messagedetail)
        messagedetailsxml.Save("messages.xml")
        Return True
    End Function
    Function messagenumber(ByVal number As Integer)
        Dim message As New System.Xml.XmlDocument
        message.Load("messages.xml")
        Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("messages/message")
```

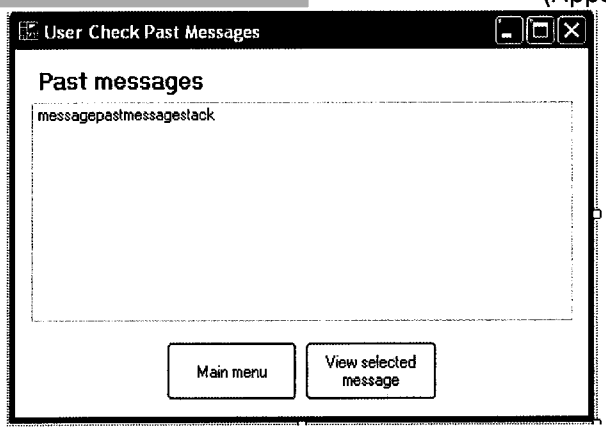
```
        messagenumber =
messagedetails.Item(number).SelectSingleNode("messagenumber").InnerText()
    End Function
    Function userid(ByVal number As Integer)
        Dim message As New System.Xml.XmlDocument
        message.Load("messages.xml")
        Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("messages/message")
        userid = messagedetails.Item(number).SelectSingleNode("userid").InnerText()
    End Function
    Function name(ByVal number As Integer)
        Dim message As New System.Xml.XmlDocument
        message.Load("messages.xml")
        Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("messages/message")
        name = messagedetails.Item(number).SelectSingleNode("name").InnerText()
    End Function
    Function datesubmitted(ByVal number As Integer)
        Dim message As New System.Xml.XmlDocument
        message.Load("messages.xml")
        Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("messages/message")
        datesubmitted =
messagedetails.Item(number).SelectSingleNode("datesubmitted").InnerText()
    End Function
    Function topic(ByVal number As Integer)
        Dim message As New System.Xml.XmlDocument
        message.Load("messages.xml")
        Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("messages/message")
        topic = messagedetails.Item(number).SelectSingleNode("topic").InnerText()
    End Function
    Function submittedmessage(ByVal number As Integer)
        Dim message As New System.Xml.XmlDocument
        message.Load("messages.xml")
        Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("messages/message")
        submittedmessage =
messagedetails.Item(number).SelectSingleNode("message").InnerText()
    End Function
    Function read(ByVal number As Integer)
        Dim message As New System.Xml.XmlDocument
        message.Load("messages.xml")
        Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("messages/message")
        read = messagedetails.Item(number).SelectSingleNode("read").InnerText()
    End Function
    Function length()
        Dim message As New System.Xml.XmlDocument
        message.Load("messages.xml")
        Dim messagedetails As System.Xml.XmlNodeList =
message.SelectNodes("messages/message")
        Return messagedetails.Count
    End Function
    Sub setmessagetoread(ByVal job As String)
        Dim messagexml = XDocument.Load("messages.xml")
        Dim message As New findjob
        Dim messagenumber As Integer
        messagenumber = message.findrightmessage(job)
        messagexml.<messages>(0).<message>(messagenumber).<read>(0).SetValue("true")
        messagexml.Save("messages.xml")
    End Sub
End Class
```

User check new message




```
Public Class usercheckknewmessages

    Private Sub mainmenu_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mainmenu.Click
        User_Menu.Show()
        Me.Close()
    End Sub

    Private Sub usercheckknewmessages_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim update As New addtoqueue
        update.addusernewmessagetostack(Login.username.Text)
    End Sub

    Private Sub viewselectedmessage_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles viewselectedmessage.Click
        Dim messageread As New reprographics_sent_messages
        Dim message As String
        message = messagestack.SelectedItem
        messageread.setmessagetoread(message)
        view_selected_new_message_user_.Show()
        Me.Close()
    End Sub

    Private Sub viewpastmessages_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles viewpastmessages.Click
        usercheckpastmessages.Show()
        Me.Close()
    End Sub
End Class
```

User check past message



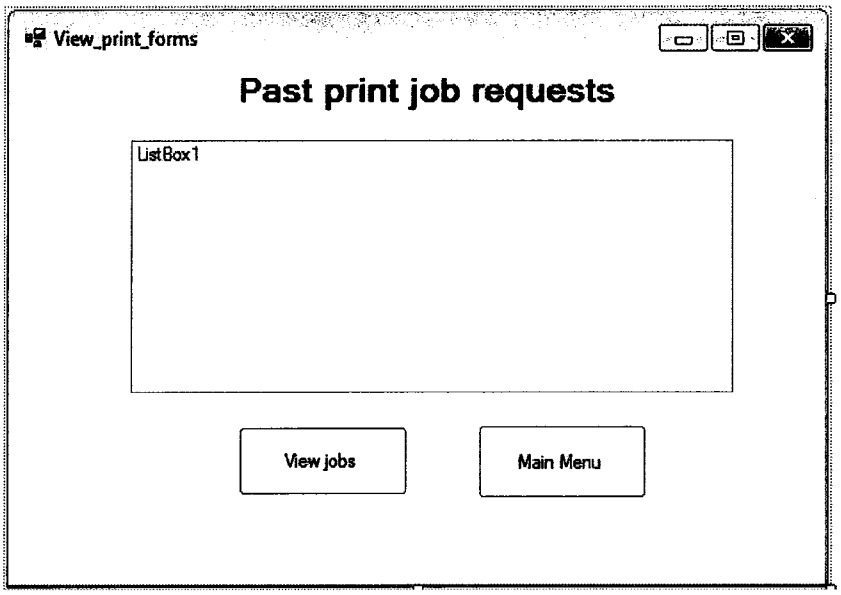

```
Public Class usercheckpastmessages

    Private Sub mainmenu_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mainmenu.Click
        User_Menu.Show()
        Me.Close()
    End Sub

    Private Sub usercheckpastmessages_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim update As New addtoqueue
        update.adduserpastmessagetostack()
    End Sub

    Private Sub viewselectedmessage_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles viewselectedmessage.Click
        view_selected_past_message__user_.Show()
        Me.Close()
    End Sub
End Class
```

User profile (class)

```
Public Class userprofile
    Function userid(ByVal number As Integer)
        Dim userdetails As New System.Xml.XmlDocument
        userdetails.Load("userdetails.xml")
        Dim userdetail As System.Xml.XmlNodeList =
userdetails.SelectNodes("users/user")
        userid = userdetail.Item(number).SelectSingleNode("userid").InnerText()
    End Function
    Function department(ByVal number As Integer)
        Dim userdetails As New System.Xml.XmlDocument
        userdetails.Load("userdetails.xml")
        Dim userdetail As System.Xml.XmlNodeList =
userdetails.SelectNodes("users/user")
        department =
userdetail.Item(number).SelectSingleNode("department").InnerText()
    End Function
    Function password(ByVal number As Integer)
        Dim userdetails As New System.Xml.XmlDocument
        userdetails.Load("userdetails.xml")
```

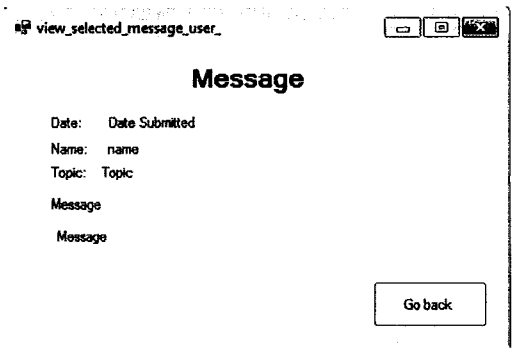
```
        Dim userdetail As System.Xml.XmlNodeList =
userdetails.SelectNodes("users/user")
        password = userdetail.Item(number).SelectSingleNode("password").InnerText()
    End Function
    Function securityquestion(ByVal number As Integer)
        Dim userdetails As New System.Xml.XmlDocument
        userdetails.Load("userdetails.xml")
        Dim userdetail As System.Xml.XmlNodeList =
userdetails.SelectNodes("users/user")
        securityquestion =
userdetail.Item(number).SelectSingleNode("securityquestion").InnerText()
    End Function
    Function securityanswer(ByVal number As Integer)
        Dim userdetails As New System.Xml.XmlDocument
        userdetails.Load("userdetails.xml")
        Dim userdetail As System.Xml.XmlNodeList =
userdetails.SelectNodes("users/user")
        securityanswer =
userdetail.Item(number).SelectSingleNode("securityanswer").InnerText()
    End Function
    Function Length()
        Dim userdetails As New System.Xml.XmlDocument
        userdetails.Load("userdetails.xml")
        Dim userdetail As System.Xml.XmlNodeList =
userdetails.SelectNodes("users/user")
        Return userdetail.Count
    End Function
    Sub changepassword(ByVal number As Integer, ByVal newpassword As String)
        Dim userxml = XDocument.Load("userdetails.xml")
        userxml.<users>(0).<user>(number).<password>(0).SetValue(newpassword)
        userxml.<users>(0).<user>(number).<confirmpassword>(0).SetValue(newpassword)
        userxml.Save("userdetails.xml")
    End Sub
    Sub saveuser(ByVal userid As Integer, ByVal department As String, ByVal password
As String, ByVal confirmpassword As String, ByVal securityquestion As String, ByVal
securityanswer As String)
        Dim userdetails = XDocument.Load("userdetails.xml")
        Dim user As XElement = _
         <user>
             <userid>[redacted] userid [redacted]</userid>
             <department>[redacted] department [redacted]</department>
             <password>[redacted] password [redacted]</password>
             <confirmpassword>[redacted] confirmpassword [redacted]</confirmpassword>
             <securityquestion>[redacted] securityquestion [redacted]</securityquestion>
             <securityanswer>[redacted] securityanswer [redacted]</securityanswer>
         </user>
        userdetails.<users>(0).Add(user)
        userdetails.Save("userdetails.xml")
    End Sub
End Class
```

View print forms



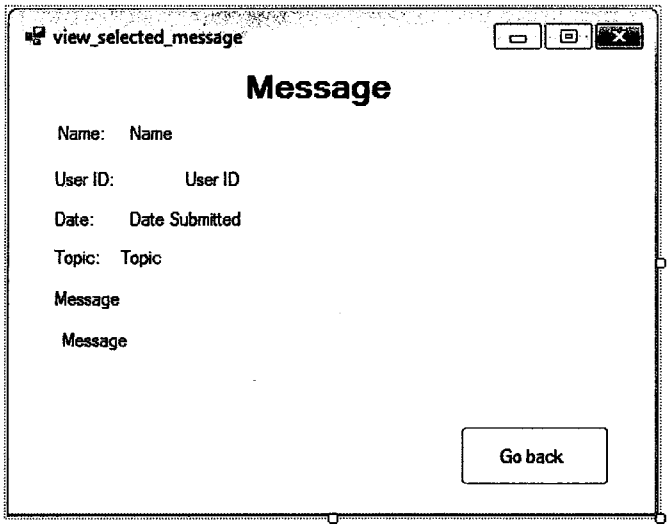

```
Public Class View_print_forms
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        User_Menu.Show()
        Me.Close()
    End Sub

    Private Sub View_print_forms_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim update As New addtoqueue
        update.addtolistpastjobs()
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        viewpastprintjobforms.Show()
        Me.Close()
    End Sub
End Class
```

View selected new message (user account)



```
Public Class view_selected_new_message_user_

    Private Sub view_selected_message_user__Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
        Dim messagedetails As New reprographics_sent_messages
        Dim number As Integer
        Dim find As New findjob
        Dim selectedjob As String
        selectedjob = usercheckmessages.messagestack.SelectedItem
        number = find.findrightmessageinuserlogin(selectedjob)
        nameentered.Text = messagedetails.name(number)
        datesubmitted.Text = messagedetails.datesubmitted(number)
        topic.Text = messagedetails.topic(number)
        message.Text = messagedetails.submittedmessage(number)
    End Sub

    Private Sub cancel_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cancel.Click
        usercheckpastmessages.Show()
        Me.Close()
    End Sub
End Class
```

View selected new message (reprographics account)



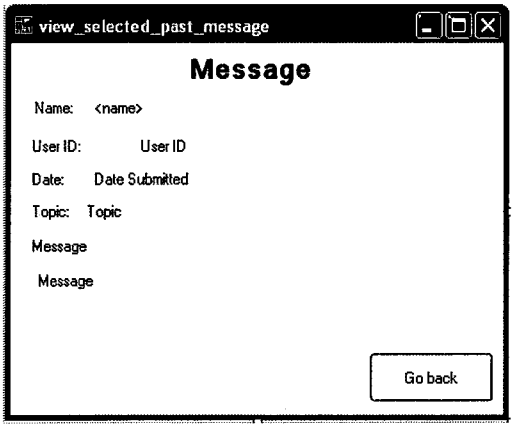

```
Public Class view_selected_new_message

    Private Sub view_selected_message_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim messagedetails As New user_sent_message
        Dim number As Integer
        Dim find As New findjob
        Dim selectedjob As String
        selectedjob = reprographics_check_messages.newmessagestack.SelectedItem
        number = find.findrightmessage(selectedjob)
        userid.Text = messagedetails.userid(number)
        nameentered.Text = messagedetails.name(number)
        datesubmitted.Text = messagedetails.datesubmitted(number)
        topic.Text = messagedetails.topic(number)
        message.Text = messagedetails.submittedmessage(number)
    End Sub

    Private Sub cancel_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cancel.Click
        Reprographics_past_messages.Show()
```

```
        Me.Close()
    End Sub
End Class
```

View selected past message (user account)


```
Public Class view_selected_past_message__user_

    Private Sub view_selected_past_message__user__Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        Dim messagedetails As New reprographics_sent_messages
        Dim number As Integer
        Dim find As New findjob
        Dim selectedjob As String
        selectedjob = usercheckpastmessages.messagepastmessagestack.SelectedItem
        number = find.findrightmessageinuserlogin(selectedjob)
        nameentered.Text = messagedetails.name(number)
        datesubmitted.Text = messagedetails.datesubmitted(number)
        topic.Text = messagedetails.topic(number)
        message.Text = messagedetails.submittedmessage(number)
    End Sub

    Private Sub cancel_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cancel.Click
        usercheckpastmessages.Show()
        Me.Close()
    End Sub
End Class
```

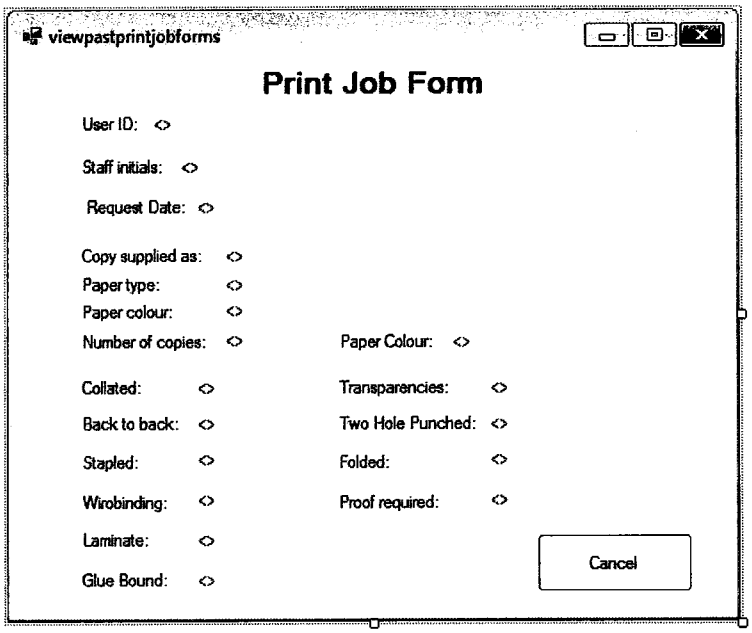
View selected past message (reprographics account)




```
Public Class view_selected_past_message
```

```
    Private Sub view_selected_past_message_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
        Dim messagedetails As New user_sent_message
        Dim number As Integer
        Dim find As New findjob
        Dim selectedjob As String
        selectedjob = Reprographics_past_messages.pastmessagesstack.SelectedItem
        number = find.findrightmessage(selectedjob)
        userid.Text = messagedetails.userid(number)
        nameentered.Text = messagedetails.name(number)
        datesubmitted.Text = messagedetails.datesubmitted(number)
        topic.Text = messagedetails.topic(number)
        message.Text = messagedetails.submittedmessage(number)
    End Sub

    Private Sub cancel_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cancel.Click
        Reprographics_past_messages.Show()
        Me.Close()
    End Sub
End Class
```

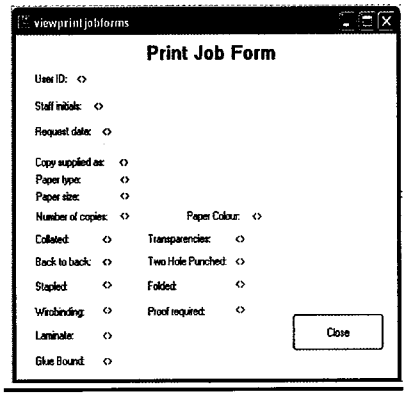
View past print job forms (user account)




```
Public Class viewpastprintjobforms

    Private Sub viewpastprintjobforms_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim viewpastjobs As New printjobform
        Dim number As Integer
        Dim jobnumber As New Print_job_queue
        Dim find As New findjob
        Dim selectedjob As String
        selectedjob = View_print_forms.ListBox1.SelectedItem
        number = find.findrightjob(selectedjob)
        number = number
        UserID.Text = viewpastjobs.userid(number)
        Staffinitials.Text = viewpastjobs.staffinitials(number)
        Requestdate.Text = viewpastjobs.requestdate(number)
```

```
        copysuppliedas.Text = viewpastjobs.copysuppliedas(number)
        papertype.Text = viewpastjobs.papertype(number)
        papersize.Text = viewpastjobs.papersize(number)
        numberofcopies.Text = viewpastjobs.numberofcopies(number)
        papercolour.Text = viewpastjobs.papercolour(number)
        backtoback.Text = viewpastjobs.backtoback(number)
        collated.Text = viewpastjobs.collated(number)
        stapled.Text = viewpastjobs.stapled(number)
        wirobinding.Text = viewpastjobs.wirobinding(number)
        laminate.Text = viewpastjobs.laminate(number)
        gluebound.Text = viewpastjobs.gluebound(number)
        transparencies.Text = viewpastjobs.transparencies(number)
        twoholepunched.Text = viewpastjobs.twoholepunched(number)
        folded.Text = viewpastjobs.folded(number)
        proofrequired.Text = viewpastjobs.proofrequired(number)
    End Sub

    Private Sub Cancel_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Cancel.Click
        View_print_forms.Show()
        Me.Close()
    End Sub
End Class
```

View print job forms (reprographics account)




```
Public Class viewprintjobforms
    Private Sub viewprintjobforms_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim viewpastjobs As New printjobform
        Dim number As Integer
        Dim jobnumber As New Print_job_queue
        Dim find As New findjob
        Dim selectedjob As String
        selectedjob = Print_job_queue.Queue.SelectedItem
        number = find.findrightjob(selectedjob)
        number = number
        UserID.Text = viewpastjobs.userid(number)
        Staffinitials.Text = viewpastjobs.staffinitials(number)
        Requestdate.Text = viewpastjobs.requestdate(number)
        copysuppliedas.Text = viewpastjobs.copysuppliedas(number)
        papertype.Text = viewpastjobs.papertype(number)
        papersize.Text = viewpastjobs.papersize(number)
        numberofcopies.Text = viewpastjobs.numberofcopies(number)
        papercolour.Text = viewpastjobs.papercolour(number)
        backtoback.Text = viewpastjobs.backtoback(number)
        collated.Text = viewpastjobs.collated(number)
```

```
        stapled.Text = viewpastjobs.stapled(number)
        wirobinding.Text = viewpastjobs.wirobinding(number)
        laminate.Text = viewpastjobs.laminate(number)
        gluebound.Text = viewpastjobs.gluebound(number)
        transparencies.Text = viewpastjobs.transparencies(number)
        twoholepunched.Text = viewpastjobs.twoholepunched(number)
        folded.Text = viewpastjobs.folded(number)
        proofrequired.Text = viewpastjobs.proofrequired(number)
    End Sub

    Private Sub closebutton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles closewindow.Click
        Print_job_queue.Show()
        Me.Close()
    End Sub

End Class
```